Red Hat

RH033

Red Hat Linux Essentials

RH033-RHEL4-2-2006-02-21

Red Hat Europe, 10 Alan Turing Road,
Guildford, Surrey. GU2 7YF.
United Kingdom
Tel: +(44)-1483-300169
FAX: +(44)-1483-574944

RH033

Red Hat Linux Essentials

RH033-RHEL4-2-20060221

Table of Contents

RH033 - Red Hat Linux Essentials

RH033: Red Hat Enterprise Linux Essentials

RH033: Red Hat Enterprise Linux Essentials

Unit 1 - Overview

Unit 2 - Running Commands and Getting Help

Copyright © 2006 Red Hat, Inc.
All rights reserved

Unit 3 - Browsing the Filesystem

Unit 4 - The bash Shell

Copyright © 2006 Red Hat Inc

Unit 5 - Standard I/O and Pipes

Unit 6 - Users, Groups, and Permissions

Copyright © 2006 Red Hat, Inc
All rights reserved.

Unit 7 - vi and vim Editor Basics

Unit 8 - The Linux Filesystem In-Depth

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Unit 9 - Configuring the Bash Shell

Unit 10 - Advanced Topics in Users, Groups and Permissions

Copyright © 2006 Red Hat, Inc
All rights reserved

Unit 11 - Advanced vi/vim and Printing

Unit 12 - Introduction to String Processing

Unit 13 - String Processing with Regular Expressions

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Unit 14 - Finding and Processing Files

Unit 15 - Investigating and Managing Processes

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Unit 16 - bash Shell Scripting

Unit 17 - Network Clients

Copyright © 2006 Red Hat Inc.
All rights reserved.

Unit 18 - So... What Now?

Copyright © 2006 Red Hat, Inc
All rights reserved.

Welcome!

RH033: Red Hat Enterprise Linux Essentials

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Welcome

Please let us know if you have any special needs while at our training facility.

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Phone and network availability

Please only make calls during breaks Your instructor will show you which phone to use. Network access and analogue phone lines may be available; your instructor will provide information about these facilities Please turn pagers to silent and cell phones off during class

Restrooms

Your instructor will notify you of the location of these facilities

Lunch and breaks

Your instructor will notify you of the areas to which you have access for lunch and for breaks

In case of Emergency

Please let us know if anything comes up that will prevent you from attending.

Access

Each facility has its own opening and closing times. Your instructor will provide you with this information.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Participant Introductions

Please introduce yourself to the rest of the class!

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Introduction

RH033: Red Hat Enterprise Linux Essentials

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Copyright

- The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2006 Red Hat, Inc.
- No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.
- This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse o contents or details contained herein.
- If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please email training@redhat.com or phone toll-free (USA) +1 866 626 2994 or +1 919 754 3700.

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Red Hat Enterprise Linux

- Enterprise-targeted operating system
- Focused on mature open source technology
- 12-18 month release cycle
  - Certified with leading OEM and ISV products
- Purchased with one year Red Hat Network subscription and support contract
  - Support available for seven years after release
  - Up to 24x7 coverage plans available

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

About Red Hat Enterprise Linux

The Red Hat Enterprise Linux product family is designed specifically for organizations planning to use Linux in the production settings. All products in the Red Hat Enterprise Linux family are built on the same software foundation, and maintain the highest level of ABI/API compatibility across releases and errata. Extensive support services are available: a one year support contract and Update Module entitlement to Red Hat Network are included with purchase. Various Service Level Agreements are available that may provide up to 24x7 coverage with guaranteed one hour response time. Support will be available for up to five years after a particular release.

Red Hat Enterprise Linux is released on a twelve to eighteen month cycle. It is based on code developed by the open source community and adds performance enhancements, intensive testing, and certification on products produced by top independent software and hardware vendors such as Dell, IBM, Fujitsu, BEA, and Oracle The longer release cycle allows vendors and enterprise users to focus on a common, stable platform and to effectively plan migration and upgrade cycles. Red Hat Enterprise Linux provides a high degree of standardization through its support for seven processor architectures (Intel x86-compatible, Intel Itanium 2, AMD AMD64/Intel EM64T, IBM PowerPC on eServer iSeries and eServer pSeries, and IBM mainframe on eServer zSeries and S/390).

Currently, on the x86-compatible architecture, the product family includes:

Red Hat Enterprise Linux AS: the top-of-the-line Red Hat Enterprise Linux solution, this product supports the largest x86-compatible servers and is available with the highest levels of support.

Red Hat Enterprise Linux ES: for entry-level or mid-range departmental servers. Red Hat Enterprise Linux ES provides the same core capabilities as AS, for systems with up to two physical CPUs and up to 8 GB of main memory.

Red Hat Enterprise Linux WS: the desktop/client partner for Red Hat Enterprise Linux AS and Red Hat Enterprise Linux ES on x86-compatible systems Based on the same development environment and same software core as the server products, Red Hat Enterprise Linux WS does not include some network server applications. It is ideal for desktop deployments or use as a compute node in a HPC cluster environment

Copyright © 2006 Red Hat, Inc.
All rights reserved

Red Hat Network

- A comprehensive software delivery, system management, and monitoring framework
  - *Update Module*: Provides software updates
    - Included with all Red Hat Enterprise Linux subscriptions
  - *Management Module*: Extended capabilities for large deployments
  - *Provisioning Module*: Bare-metal installation, configuration management, and multi-state configuration rollback capabilities

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Red Hat Network

Red Hat Network is a complete systems management platform. It is a framework of modules for easy software updates, systems management, and monitoring, built on open standards. There are currently four modules in Red Hat Network; the Update Module, the Management Module, the Provisioning Module, and the Monitoring Module.

The Update Module is included with all subscriptions to Red Hat Enterprise Linux. It allows for easy software updates to all your Red Hat Enterprise Linux systems

The Management Module is an enhanced version of the Update Module, which adds additional functionality tailored for large organizations These enhancements include system grouping and set management, multiple organizational administrators, and package profile comparison among others. In addition, with RHN Proxy Server or Satellite Server, local package caching and management capabilities become available.

The Provisioning Module provides mechanisms to provision and manage the configuration of Red Hat Enterprise Linux systems throughout their entire life cycle. It supports bare metal and existing state provisioning, storage and editing of kickstart files in RHN, configuration file management and deployment, multi-state rollback and snapshot-based recovery, and RPM-based application provisioning. If used with RHN Satellite Server, support is added for PXE bare-metal provisioning, an integrated network tree, and configuration management profiles.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Red Hat Desktop

- High-quality, full-featured client system based on Red Hat Enterprise Linux
  - Includes desktop productivity applications
- Available in packages of 10 or 50 units for mass deployments of desktop systems
- Clients entitled to RHN Management Module
  - Package may also include RHN Proxy Server or Satellite Server

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Red Hat Desktop

Red Hat Desktop is the latest addition to the Red Hat Enterprise Linux product family. It provides a high-quality, full-featured client system suitable for user in a wide range of desktop deployments. Red Hat Desktop includes integrated third party applications including Adobe Acrobat Reader and plugin, Macromedia Flash plugin, Citrix ICA client, Java (IBM and BEA) and plugin (IBM), and Real Player. Red Hat Desktop shares the same primary product features as the rest of the Red Hat Enterprise Linux family, including a twelve to eighteen month release cycle and one year of bundled software updates and support (annually renewable for the life of the product).

Red Hat Desktop provides mechanisms to help manage and secure large desktop deployments. It is available in packages of either 10 or 50 units for mass deployment of consistently managed clients. Client systems are bundled with Red Hat Network Management Module entitlements for improved manageability. In addition, packages may include either Red Hat Network Proxy or Red Hat Network Satellite Server (with a Red Hat Enterprise Linux AS, Premium Edition entitlement) to ensure the highest levels of manageability and security.

Red Hat Desktop is currently available for client systems based on either the Intel x86 or the AMD AMD64/Intel EN64T processor architecture with a single CPU and up to 4 GB of main memory.

Red Hat Applications

- Open source applications provided separately from Red Hat Enterprise Linux
- Include a range of support options
- Installation media and updates provided through Red Hat Network
- More information on specific products at `http://www.redhat.com/software/rha/`

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Red Hat Applications

Red Hat offers a number of additional open source application products and operating system enhancements which may be added to the standard Red Hat Enterprise Linux operating system As with Red Hat Enterprise Linux, Red Hat provides a range of maintenance and support services for these add-on products. Installation media and software updates are provided through the same Red Hat Network interface used to manage Red Hat Enterprise Linux systems The Red Hat Applications product family includes software for high availability clusters of Linux systems, software development, and management of web content

For more information on specific products which are currently available, please visit `http://www.redhat.com/software/rha/`

Copyright © 2006 Red Hat, Inc.
All rights reserved.

The Fedora Project

- Read Hat sponsored open source project
- Focused on latest open source technology
  - Rapid four to six month release cycle
  - Available as free download from the Internet
- An open, community-supported proving ground for technologies which may be used in upcoming enterprise products
- Red Hat does not provide formal support

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

About the Fedora Project

The Fedora Project is a community supported open source project sponsored by Red Hat intended to provide a rapidly evolving, technology-driven Linux Distribution with an open, highly scalable development and distribution model. It is designed to be an incubator and test bed for new technologies that may be used in later Red Hat enterprise products. The basic Fedora Core distribution will be available for free download from the Internet.

The Fedora Project will produce releases on a short four to six month release cycle, to bring the latest innovations of open source technology to the community. This may make it attractive for power users and developers who want access to cutting-edge technology and can handle the risks of adopting rapidly changing new technology. Red Hat does not provide formal support for the Fedora Project.

For more information, visit `http://fedora.redhat.com`

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Classroom Network

	Names	IP Addresses
Our Network	example.com	192.168.0.0/24
Our Server	server1.example.com	192.168.0.254
Our Stations	stationX.example.com	192.168.0.X
Hostile Network	cracker.org	192.168.1.0/24
Hostile Server	server1.cracker.org	192.168.1.254
Hostile Stations	stationX.cracker.org	192.168.1.X
Trusted Station	trusted.cracker.org	192.168.1.21

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Objectives

- A Red Hat Enterprise Linux user who can be productive in using and customizing his or her own Red Hat Enterprise Linux system for common command-line processes and desktop productivity roles

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

A person who has completed RH033 will have practiced the basic skills required to be a productive Red Hat Linux user. These skills include:

- File and directory operations
- Understanding users and groups
- Standard I/O and pipes
- String processing
- Managing processes
- bash shell operations
- The graphical environments in Red Hat Enterprise Linux
- Printing and Mail
- Basic networking tools ping, trace
- The vi editor
- System tools
- Red Hat Linux installation

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Audience and Prerequisites

- Audience: Users new to Linux and UNIX; users and administrators transitioning from another operating system
- Prerequisites: User-level experience with any computer system; use of mouse, menus, and any graphical user interface

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Unit 1

Overview

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Objectives

Upon completion of this unit, you should:

- Understand UNIX/Linux history and principles
- Be familiar with Open Source licenses
- Understand the significance of Red Hat Enterprise Linux
- Be able to use Linux's graphical environments

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

UNIX History

- First version created in Bell Labs - 1969
- AT&T licenses source code for low cost
  - Trademarks UNIX name, "UNIX" name closely held
    - Licensees must create new name for their operating systems
  - Many UNIX "flavors" emerge

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Bell Labs

UNIX was originally developed for internal use at AT&T by researchers Ken Thompson and Dennis Ritchie AT&T licensed the source code, widely allowing many companies to modify and produce UNIX-like operating systems Because AT&T held the UNIX name, other companies had to create their own names to brand the modifications and additions they had made: AIX from IBM, HP/UX from Hewlett-Packard, SunOS (later Solaris) from Sun, and IRIX from SGI

UNIX flavors

These many flavors of UNIX operate in a similar manner. At the shell prompt, most offer the same standard utilities and commands, although the parameters a command uses may vary from system to system. One can compare UNIX to cars; there are many different makes and models of cars, but fundamentally they all work and are operated the same way, though there are minor differences

Copyright © 2006 Red Hat, Inc.
All rights reserved

UNIX Principles

- Everything is a file (Including hardware)
- Configuration data stored in text
- Small, single-purpose programs
- Avoid captive user interfaces
- Ability to chain programs together to perform complex tasks

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Everything is a file

UNIX systems have many powerful utilities designed to create and manipulate files The UNIX security model is based around the security of files By treating everything as a file, a consistency emerges. You can secure access to hardware in the same way as you secure access to a document

Configuration data stored in text

Text is a universal interface, and many UNIX utilities exist to manipulate text. Storing configuration in text allows an administrator to move a configuration from one machine to another easily. There are several revision control applications that enable an administrator to track which change was made on a particular day, and provide the ability to roll back a system configuration to a particular date and time.

Small, single-purpose programs

UNIX provides many small utilities that perform one task very well. When new functionality is required, the general philosophy is to create a separate program - rather than to extend an existing utility with new features

Avoid captive user interfaces

Interactive commands are rare in UNIX Most commands expect their options and arguments to be typed on the command line when the command is launched. The command completes normally, possibly producing output, or generates an error message and quits Interactivity is reserved for programs where it makes sense, for example, text editors (of course, there are non-interactive text editors too.)

Ability to chain programs together to perform complex tasks

A core design feature of UNIX is that the output of one program can be the input for another. This gives the user the flexibility to combine many small programs together to perform a larger, more complex task.

Copyright © 2006 Red Hat, Inc
All rights reserved

GNU Project / FSF

- GNU Project started in 1984
  - Goal: Create a "free" UNIX clone
  - By 1990, nearly all required userspace applications created
    - **gcc**, **emacs**, etc

/usr/bin

- Free Software Foundation
  - Non-profit organization that manages the GNU project

Richard Stallman

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

What is free software?

The term "free software" may have a different meaning than you expect The term doesn't refer to the cost of the software, but the fact that end user has the freedom to modify and change the program. The GNU web site reads, in part:

> To understand the concept, you should think of "free speech", not "free beer "
>
> "Free software" refers to the users' freedom to run,copy, distribute, study, change and improve the software More precisely, it refers to four kinds of freedom, for the users of the software:
>
> The freedom to run the program, for any purpose (freedom 0).
>
> The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
>
> The freedom to redistribute copies so you can help your neighbor (freedom 2).
>
> The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom3) Access to the source code is a precondition for this.
>
> A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere.

(http://www.gnu.org/philosophy/free-sw.html)

GPL - GNU General Public License

- Primary license for Open Source software
- Encourages free software
- Enhancements and changes to GPL software must also be GPL
- Often called "copyleft"
  - "All rights reversed"

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Open Source Licenses

Most of the utilities and applications included with Red Hat Linux are also covered by the GPL. One major exception is the X Window System, which has its own Terms and Conditions. The text of the GPL can be found at http://www.gnu.org/copyleft/gpl.html

A few applications have their own licensing agreements, which must be agreed to before they can be used.

Software developers (and others) who use code from BSD will need to agree to abide by the terms of the Berkeley Software Distribution. Visit http://www bsd.com for more information

All of the software contained in Red Hat Linux is free for end users. However, if you are going to be developing commercial applications, or wish to redistribute some of the software in the distribution, read the appropriate licenses and agreements first.

can sell but must supply source code

Copyright © 2006 Red Hat, Inc
All rights reserved

Linux Origins

- Linus Torvalds
  - Finnish college student in 1991
  - Created Linux kernel
- Linux kernel + GNU applications = complete, free, UNIX-like OS

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The beginnings of Linux

Linus Torvalds announced Linux in the comp os minix newsgroup in August, 1991:

From: torvalds@klaava Helsinki FI (Linus Benedict Torvalds)

Newsgroups: comp os minix

Subject: What would you like to see most in minix?

Summary: small poll for my new operating system

Message-ID: <1991Aug25 205708 9541@klaava.Helsinki.FI>

Date: 25 Aug 91 20:57:08 GMT

Organization: University of Helsinki

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things)

I've currently ported bash(1 08) and gcc(1.40), and things seem to work This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(

Copyright © 2006 Red Hat, Inc.
All rights reserved

Why Linux?

- Fresh implementation of UNIX APIs
- Open Source development model
- Supports wide variety of hardware
- Supports many networking protocols and configurations
- Fully supported

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Linux is a UNIX-like OS

Linux is as similar to UNIX as the various UNIX versions are to each other. Conceptually, anything that can be done with another version of UNIX can be done with the Linux operating system, although the means may vary slightly.

Multi-user and multi-tasking

Linux is a multi-user and multi-tasking operating system That means that more than one person can be logged on to the same Linux computer at the same time. (Of course, each user needs his own terminal.) The same user could even be logged into their account from two or more terminals at the same time. Linux is also multi-tasking: a user can have more than one process (program) executing at the same time

Wide hardware support

Red Hat Linux supports most pieces of modern x86-compatible PC hardware. In the early days of Linux, hardware support was limited; today, Linux support has become a checklist item for hardware vendors.

Fully supported

Red Hat Linux is a fully supported distribution. Red Hat Inc. provides many support programs for the smallest to the largest companies

Copyright © 2006 Red Hat, Inc.
All rights reserved

Red Hat Enterprise Linux

- A *distribution* of Linux
  - Custom version of a recent Linux kernel
  - Utilities and applications
  - Installation and configuration software
  - Support available

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

What is Red Hat Enterprise Linux?

Red Hat Enterprise Linux (RHEL) is a collection of software packages, including network services, applications and installation and configuration software, all based around a rigorously tested, recent version of the Linux kernel. Such software collections are called "distributions" of Linux

One of Linux's strengths lays in its versatility There are hundreds of active Linux distributions, each of which survives by adding value to the base operating system for a particular niche audience

Red Hat adds value for Enterprise customers through packaging, engineering, and quality assurance. Red Hat backs up Red Hat Enterprise Linux with comprehensive technical support and a complete range of services, training, and consulting. Because of the enormous number of RHEL systems in use around the world, Red Hat acts as a standardizing force in the diverse Linux community

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Recommended Hardware Specifications

- Pentium Pro or better with 256 MB RAM *or*
- 64-bit Intel/AMD with 512 MB RAM
- 2-6 GB disk space
- Bootable CD
- Other processor architectures supported
  - Itanium 2, IBM Power, IBM Mainframe

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

What you will need to Run Red Hat Enterprise Linux

Linux can run on very modest hardware such as 386 processors with only a few MB of RAM. Red Hat Enterprise Linux, however, is optimized for more powerful hardware. The code shipped on RHEL is compiled for Pentium Pro or better processors (as well as some other enterprise-level architectures) and will not run on older CPUs. While it might run on relatively limited hardware, RHEL will perform optimally on systems with faster processors, more RAM, and larger/faster disk drives.

In some cases, warning or error messages may appear during the install process telling the user that the hardware setup is either unsupported (due to limited RAM size for instance) or just not compatible. Incompatibility is typically caused by processors for which Red Hat's code has not been compiled, such as an original Pentium or older processor.

For systems incorporating the X Window System GUI, a mouse is also required and systems that are part of a network will need a network card and/or a modem

The recommended hardware required by RHEL is relatively minimal, but it is important to consider the task at hand. While a 800 MHz Pentium III machine (which meets the minimum requirements) could function quite nicely in a small office environment, a large scale operation serving hundreds or thousands of users will certainly require more powerful hardware.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Local Logins

- Text-mode login at virtual console
- Graphical login

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Logging in

Access to a system requires authentication The most common method used to authenticate a user is a *login* process, by entering a valid user name and password When logging in at the system console, you are presented either a text-based or graphical login In either case, you'll be presented with a prompt for your login name, followed by a prompt for your password. If either is entered incorrectly, your login will be rejected. If both are entered correctly, you will be logged in

What you see next will depend upon a number of things If your system is text-based, you will have a command prompt, probably ending in a dollar sign ($) For example:

```
localhost login: bob
Password:
Last login: Mon Sep 15 19:30:46 on :0
[bob@localhost bob]$
```

Notice that your password is not displayed when typed

On systems that boot directly into The X Window System, what you see depends on the display manager being used. The default display manager for Red Hat Linux is **gdm**, the GNOME Display Manager

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Virtual Consoles

- Multiple non-GUI logins are possible through the use of virtual consoles
- There are by default 6 available virtual consoles
- Available through *Ctrl-Alt-F[1-6]*
- If X is running, it is available as *Ctrl-Alt-F7*

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Virtual Consoles

The virtual consoles enable a user to have multiple logins even when not using an X window system. They provide full screen, non-GUI access to the system The virtual console available through *Ctrl-Alt-F7* runs the X window system when X is running.

You can scroll at the virtual consoles by using *Shift-PgUp* and *Shift-PgDn*. Be aware that the scroll buffer is stored in video memory, so if you change virtual consoles, the scroll buffer will be lost.

Copyright © 2006 Red Hat, Inc.
All rights reserved

The Xorg GUI Framework

- Modern, free implementation of XFree86
- Highly flexible framework for displaying graphical applications and environments
- Completely network-transparent client/server architecture
- System can be configured to present a graphical login screen on *Ctrl-Alt-F7*

X11 server

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The X.Org Framework Just draws clients (own or other)

The X.Org system is the foundation for the graphical user interface(GUI) on Red Hat Enterprise Linux. It is based on the older Xfree86 framework, in use on most flavors of Unix, and maintained by the X Consortium at http://www.x.org. The goals of X.Org include a faster and more open development model than Xfree86, support for a wide variety of video cards and input devices and the development of a highly modular and flexible graphical framework for Unix and Linux. More information on the project is available at http://xorg.freedesktop.org

It is important to understand X's relationship to what you see on the screen. X does not define how anything should look or behave. Instead, X focuses on providing a standard way in which applications, called X clients, may display, or "write" on the screen. The X server is the program that speaks through your video hardware. Any application that wants to communicate through the display is an X client. The visual effects of a mouse cursor (an arrow, or pointing hand) selecting a link on a web page are X client activities that spawn X server events that inform the web browser to send an HTTP request to the link's target(or "anchor"). You do not really see the X server, but X clients. X provides the data I/O infrastructure for X clients, like a the human nervous system, it sends messages when "touched" by client activity.

Originally designed as a client and server application suite, X11 uses UNIX-domain or TCP/IP networking for its operation, where one server provides many clients--both hardware(hosts and displays) and software(applications and widgets)--a *protocol* through which to pass data. Expressed in this design are two layers: a device dependent, hardware layer, and a device independent software layer. The hardware layer manages the coordination of mouse and keyboard(input) and video card and display(output). The software layer provides an API as the basis of uniform visual characteristics and rendering across varied platforms. The combination of both layers provides X client applications greater hardware and operating system independence. Also, an X client running on one system can display on any X server running on any operating system, *if* sufficient access is granted. On a single workstation, the X clients and X server still communicate via the X protocol, but instead of using TCP/IP, they use

Copyright © 2006 Red Hat, Inc
All rights reserved.

a high-speed Unix domain socket. For each managed display, this socket is `/tmp/.X11-unix/X#` (where # is 0 to the greatest number of permitted connections)

Copyright © 2006 Red Hat, Inc.
All rights reserved.

The Xorg Graphical Environments

- Collections of applications that provide a graphical working environment with a consistent look-and-feel
  - GNOME - The default desktop environment
  - KDE - Environment based on the Qt toolkit

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Consistent User interfaces

GNOME (GNU Network Object Model Environment) is a user-friendly desktop environment that allows users to easily configure and use their computers GNOME includes a panel along the bottom of the screen for launching applications and displaying information. GNOME also includes a set of applications and desktop tools. GNOME is a set of standards and conventions that can be used by applications so that they can communicate and be consistent with each other.

KDE is another user-friendly desktop environment provided with Red Hat Linux It's easy to configure both GNOME and KDE on your Red Hat Linux workstation, and to switch between the two desktops depending on your preference

Copyright © 2006 Red Hat, Inc
All rights reserved.

Starting Xorg

- Nothing needed if system boots to a graphical login. Just authenticate.
- If system boots to a virtual console login, Xorg must be started manually
  - Run **startx** to manually start Xorg

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Consistent User interfaces

GNOME (GNU Network Object Model Environment) is a user-friendly desktop environment that allows users to easily configure and use their computers. GNOME includes a panel along the bottom of the screen for launching applications and displaying information. GNOME also includes a set of applications and desktop tools GNOME is a set of standards and conventions that can be used by applications so that they can communicate and be consistent with each other

KDE is another user-friendly desktop environment provided with Red Hat Linux It's easy to configure both GNOME and KDE on your Red Hat Linux workstation, and to switch between the two desktops depending on your preference

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Changing Your Password

- Passwords should be changed after first login
- In Gnome: Applications->Preferences->Password
- From a terminal: **passwd**

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Choosing a good password

When you first receive your account, a password will be provided so you can log on. You should immediately change your password! Remember that Linux passwords are case-sensitive.

Valid passwords should adhere to the following rules:

- at least 6 but no more than 255 characters
- contain at least one non-alphanumeric character
- not be based on a dictionary word
- not be the similar to the current password
- not be similar to the login id

Before the password can be changed, the current password must be supplied. Of course, any password will not be displayed. You will have to enter your new password twice to ensure that you didn't make a typing mistake.

Your password can also be changed from within the Red Hat desktop. From the Applications menu, select Preferences->Password

If you enter an invalid password, the system will return an error message and allow you to try again

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 1

- Questions and Answers
- Summary
  - UNIX/Linux History
  - The GNU General Public License
  - Red Hat Enterprise Linux

17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Unit 2

Running Commands and Getting Help

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Objectives

Upon completion of this unit, you should:

- Know how to execute commands at the prompt
- Be familiar with some simple commands
- Be familiar with help resources in Red Hat Enterprise Linux

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Running Commands

- Commands have the following syntax:
  - **command** *`options arguments`*
- Each item is separated by a space
- Options modify a command's behavior
  - Single-letter options usually preceded by **-**
    - Can be passed as **-a -b -c** or **-abc**
  - Full-word options usually preceded by --
    - Example: **--help**
- Arguments are filenames or other data needed by the command
- Multiple commands can be separated by ;

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Interrupting command execution

If you enter a command and don't get a prompt back, the command may be busy executing or may be waiting for input

To interrupt a command taking too long to execute, type *Ctrl-c*.

Occasionally you might enter a command without an argument that expects input to come from the keyboard. In this case, type *Ctrl-c* to terminate the command (More on this in Unit 8)

Running multiple commands

You can separate multiple commands on the same line with semicolons. When the first command finishes, the next one will execute

```
[student@stationX ~]$ mkdir backups; cp *.txt backups/
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

Some Simple Commands

- **date** - display date and time
- **cal** - display calendar

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

date prints the system time and date. The format is configurable via an optional formatting string (see options with **date --help**) The example below demonstrates this feature.

```
[student@stationX ~]$ date
Sun May  5 14:57:05 EDT 2002
[student@stationX ~]$ date +"Today is %A, %B %d, %Y.%nIt is %r, %Z."
Today is Friday, January 06, 2006.
It is 12:05:41 PM, EST.
```

cal prints an ASCII calendar of the current month. When given a single numeric argument, **cal** will give a calendar for the given year. Use a four digit year, however, as the command **cal 06** will give a calendar for the year 06, not the year 2006.

Given a month and year as arguments, **cal** will display the calendar for that particular month. For example:

```
[student@stationX ~]$ cal 9 2010
  September 2010
Su Mo Tu We Th Fr Sa
          1  2  3  4
5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```

Try displaying the calendar for January, 1752

Copyright © 2006 Red Hat, Inc.
All rights reserved

Getting Help

- Don't try to memorize everything!
- Many levels of help
  - **whatis**
  - *`command`* **--help**
  - **man** and **info**
  - `/usr/share/doc/`
  - Red Hat documentation

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

For all its advantages, one disadvantage of the command-line interface is the large number of commands and arguments that the user must take advantage of to use it well. A common mistake made by people new to the command line is to assume that this requires every little argument to be committed to memory.

In order to be proficient one needn't become a walking database of Linux arcana (though that does tend to come with time) In fact, while committing regularly-used commands and arguments to memory is always helpful, the key to effectively using a command-line operating system is the ability to use the available resources to quickly look up arguments and techniques that you don't know by heart The following slides will discuss several such resources

Copyright © 2006 Red Hat, Inc.
All rights reserved

The whatis Command

- Displays short descriptions of commands
- Uses a database that is updated nightly
- Often not available immediately after install

```
$ whatis cal
cal      (1)  - displays a calendar
```

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

In your explorations of a Red Hat Enterprise Linux system you will periodically encounter a "mystery command"; that is, a command that looks interesting though you are not entirely sure what it does One way to find out what a command does is, of course, to run it but this can be risky. The **whatis** tool provides an easier (and safer) way of getting a quick explanation of what another command does

whatis accepts the name of another command as its only argument It then searches for the given command name in a database of short descriptions If it finds a match, the description is printed to your screen Along with the description, **whatis** prints the command's name and a number in parenthesis. This number represents the "chapter" of the Linux Manual where more thorough documentation can be found Forthcoming slides will explain the Linux Manual and the associated **man** command in more detail.

The database that **whatis** uses is (re)generated automatically every night. This means that on newly-installed systems **whatis** will not work at first because the database does not yet exist. The impatient can generate a database without waiting for the automatic update by asking an administrator to log in and run **makewhatis** as root.

Copyright © 2006 Red Hat, Inc.
All rights reserved

The --help Option

- Displays usage summary and argument list
- Used by most, but not all, commands

```
$ date --help
Usage: date [OPTION]... [+FORMAT] or:
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
Display the current time in the given FORMAT,
or set the system date.
 ...argument list omitted...
```

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Just knowing what a command does isn't always enough In order to use a command effectively you need to know what options and arguments it accepts and what order it expects them in (the *syntax* of the command) Most commands have a --help option This causes the command to print a description of what it does, a "usage statement that describes the command's syntax and a list of the options it accepts and what they do

At first glance, usage statements may seem complicated and difficult to read However, they become much simpler to understand once one is familiar with a few basic conventions:

- Anything in straight braces ([]) is optional
- Anything followed by ... represents an arbitrary-length list of that thing
- If you see multiple options separated by pipes (|) it means you can use any one of them
- Text in straight brackets (< >) represents variable data So `<filename>` means "insert the filename you wish to use here". Sometimes, as in the example in the slide, such variables are simply written in all caps

So, looking at the first usage statement for the **date** command:

```
date [OPTION]... [+FORMAT]
```

we see that date can take an optional list of options (`[OPTION]...`) followed by an optional format string, prefixed with a +, that defines what you want the date to look like (`[+FORMAT]`) Since both of these are optional, you will note that **date** will work even if it is not given options or arguments (it will print the current date and time using its default format).

Reading Usage Summaries

- Printed by **--help**, **man** and others
- Used to describe the syntax of a command
  - Arguments in `[]` are optional
  - Arguments in CAPS or `<>` are variables
  - Text followed by `...` represents a list
  - `x|y|z` means "`x` or `y` or `z`"
  - `-abc` means "any mix of `-a`, `-b` or `-c`"

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The man Command

- Provides documentation for commands
- Almost every command has a man "page"
- Pages are grouped into "chapters"
- Collectively referred to as the Linux Manual
- **man [<chapter>] <command>**

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Almost every command (as well as most configuration files and several developer's libraries) on a Red Hat Enterprise Linux system has an associated man page, which provides more thorough documentation than the --help option. Man pages normally contain sections discussing the following aspects of a command's usage:

- Its NAME and a short description of what it does
- A SYNOPSIS of its usage, including available switches
- A longer DESCRIPTION of the command's functionality
- A switch-by-switch listing of its OPTIONS
- Any FILES associated with this command
- Any known BUGS in the command
- EXAMPLES, showing how to use the command
- A SEE ALSO section for further reference

The collection of all man pages on a system is called the Linux Manual. The Linux Manual is divided into sections, each of which covers a particular topic, and every man page is associated with exactly one of these sections. The sections are:

Manual sections

1	User commands	4	Special files	7	Miscellaneous
2	System calls	5	File formats	8	Administrative commands
3	Library calls	6	Games		

Often, Linux commands, calls, and files are referenced by a name followed by manual section number in parentheses. For example, `passwd(1)`, which is accessed by running **man 1 passwd**, refers to the user

Copyright © 2006 Red Hat, Inc.
All rights reserved

command **passwd**, whereas passwd(5), accessed by running **man 5 passwd**, refers to the file format for /etc/passwd.

Navigating man Pages

- While viewing a man page
  - Navigate with arrows, *PgUp*, *PgDn*
  - `/text` searches for text
  - `n`/`N` goes to next/previous match
  - `q` quits
- Searching the Manual
  - **man -k** *`keyword`* lists all matching pages
  - Uses **whatis** database

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Knowing how to efficiently navigate and search a man page will save you enormous amounts of frustration and make you a much more effective Linux user. To search for text in a man page, simply type a forward slash (/) followed by the term you are looking for and press *Enter*. This will highlight ever instance of the searched-for word and take your cursor to the first found instance. Pressing `n` and `N` will move you forward and backward one match, respectively. When you are done viewing using **man**, press `q` to quit. Later in this class you will learn about a command called **less**, which is used for displaying and navigating large amounts of text one page at a time and uses the same commands for moving and searching. This similarity is not coincidental. In fact, when you view a man page, **man** transparently uses **less** to display the page.

But what if you don't know the name of the command you are looking for? You can do keyword searches and list all commands whose short descriptions match the specified keyword using **man**'s **-k** option. Note that this uses the **whatis** database, discussed earlier, and so will not be immediately available after a fresh install.

Copyright © 2006 Red Hat, Inc.
All rights reserved

The info Command

- Similar to **man**, but often more in-depth
- Run **info** without args to list all page
- **info** pages are structured like a web site
  - Each page is divided into "nodes"
  - Links to nodes are preceded by *
  - **info** [*command*]

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Many commands, notably the GNU utilities, include an info page to supplement, and in some cases replace, their man pages While man pages are usually written as a quick references, rather than thorough introductions to commands, info pages are often much more verbose and go into more detail (though sometimes they are just copies of the corresponding man page).

The structure of an info page is similar to that of a website. When the info page for a command is opened, the reader is presented with the *top node* of the info page. The top node usually contains a summary of what the command does followed by a main menu Each item in the menu is a link to another node of the page Links are denoted by a preceding asterisk (*).

If you run info with no arguments you will be presented with a list links to the top nodes of every available info page.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Navigating info Pages

- While viewing an **info** page
  - Navigate with arrows, *PgUp*, *PgDn*
  - *Tab* moves to next link
  - `Enter` follows the selected link
  - **n/p /u** goes to the next/previous/up-one node
  - `s text` searches for text (default: last search)
  - `q` quits **info**

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

To follow a link within an info page, simply position your cursor over the link and press *Enter*. You can navigate using the arrow keys or jump from link to link with the *Tab* key. At the top of your screen you will see text indicating the name of your current node, the next and previous nodes in the tree and the node directly "above" the current one, usually the node you linked from. You can reach these by pressing the **n**, **p** or **u** keys respectively.

To search the text of the current node, press the **s** key, type the term you are looking for and press *Enter*. The next time you type **s**, you will see that the last term you searched for is selected by default, so to search again simply press *Enter*. If you prefer the navigation keys used by **man**, such as **/**, **n** and **N**, you can start **info** with the **vi-keys** argument.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Extended Documentation

- The `/usr/share/doc` directory
  - Subdirectories for most installed packages
  - Location of docs that don't fit elsewhere
    - Example configuration files
    - Html/pdf/ps documentation
    - License details

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Applications often include documentation that does not fit the length or the format of a man or info page While sometimes this will be as simple as a copy of the software's license, it can also contain sample configuration files, tutorials and even entire books extending the documentation of the application.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Red Hat Documentation

- Available on docs CD or Red Hat website
  - Installation Guide
  - Intro to System Administration
  - System Administration Guide
  - Reference Guide
  - Security Guide
  - Step-by-Step Guide
  - ...and more

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Red Hat Enterprise Linux includes a great deal of documentation outside of the material provided by individual applications. These docs are available in html format and as RPM packages on the RHEL Documentation CD and at Red Hat's website: `http://www.redhat.com/docs/`.

They cover a range of topics from introductory levels to advanced and, unlike more general documentation available on the Internet, are written specifically for Red Hat Enterprise Linux.

Copyright © 2006 Red Hat, Inc.
All rights reserved

End of Unit 2

- Questions and Answers
- Summary
  - Running Commands
  - Getting Help

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc
All rights reserved

Lab 2
Getting Help with Commands

Goal:	Become familiar with the resources available to you for answering questions about Red Hat Enterprise Linux commands
Estimated Duration:	10 minutes
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named student with a password of student

Copyright © 2006 Red Hat, Inc.
All rights reserved

Sequence 1: Using the Help Tools

Instructions:

1. Look at the command-line options for the **man** command What **man** option can be used to search the name of every manual page for a keyword and list the matches (the same behavior as **whatis**)?

 ____

2. What **man** option can be used to search the name and short description of every manual page for a keyword and list the matching pages?

 ____

3. What **man** option can be used to search the entire text (not just the names and short descriptions) of the manual for a keyword, displaying the matching pages one at a time?

 ____

4. Suppose you wanted to view the man page for the `basename` function of the C programming language, as opposed to that of the **basename** command How might you do that?

 HINT: C functions are discussed in chapter 3 of the manual

 ____

5. What command-line options might you use to cause **ls** to display a long listing of files with human-readable size descriptions (ie 6.8M instead of 6819467)?

 HINT: You will need two command-line options

 ____

Copyright © 2006 Red Hat, Inc.
All rights reserved.

6. Given the usage description below, which of the following would be a syntactically valid invocation of the command **foo**?

```
foo -x|-y -[abcde] FILENAME...
```

1. **foo -x -y -a one.txt**

2. **foo**

3. **foo -y -abc one.txt two.txt**

4. **foo -abc one.txt two.txt three.txt**

Copyright © 2006 Red Hat, Inc
All rights reserved

Sequence 1 Solutions

1. **man -f *keyword*** returns a list of all man pages that contain *keyword* in the name, which is the same thing that **whatis *keyword*** would do.

2. **man -k *keyword*** will list all manual pages that contain *keyword* in the name or short description.

3. **man -K *keyword*** will display every man page that contains *keyword* For each matching page the title will be displayed and you will be asked whether or not you would like to read it.

4. **man 3 basename** would display the man page for the `basename()` function from chapter 3 of the Red Hat Enterprise Linux manual. Just running **man basename** would have displayed the page for the **basename** command, since it appears before `basename()` in chapter 1.

5. **ls -lh *filename*** would display a long listing (**-l**) with human-readable (**-h**) sizes.

6. This usage summary requires exactly one of **-x** or **-y**, followed by zero or more of **-a**, **-b**, **-c**, **-d** or **-e** followed by one or more filenames. Thus,

   foo -y -abc one.txt two.txt

   is the correct form of the command

Copyright © 2006 Red Hat, Inc
All rights reserved.

Unit 3

Browsing the Filesystem

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Know important elements of the filesystem hierarchy
- Be able to copy, move, and remove files
- Be able to create and view files
- Know how to manage files with Nautilus

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Linux File Hierarchy Concepts

- Files and directories are organized into a single-rooted inverted tree structure
- Filesystem begins at the *root* directory, represented by a lone / (forward slash) character.
- Names are case-sensitive
- Paths are delimited by /

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Filesystem Basics

These Linux file hierarchy concepts will be expanded upon in the pages that follow

- Files and directories are organized into a single-rooted inverted-tree structure, including distinct physical volumes such as floppy disks, CD-ROMs and multiple hard drives.
- The base of the inverted-tree hierarchy is known as *root* or / - the top of the file structure.
- A forward slash separates elements of a pathname, for example `/usr/bin/X11/X`
- Names in the Linux file hierarchy are case-sensitive.
- Each shell and process on the system has a designated *current* or *working* directory.
- `..` refers to the parent directory of any particular directory - one level up in the file hierarchy.
- `.` refers to the current directory.
- Files and directories whose names begin with a `.` are *hidden* -- that is, they are not displayed by default in filename listings
- A user's *path* is a list of directories that are searched for commands typed at the command line.

Some Important Directories

- The home directories
  - `/root`, `/home/`*`username`*
- The bin directories
  - `/bin`, `/usr/bin`, `/usr/local/bin`
  - `/sbin`, `/usr/sbin`, `/usr/local/sbin`
- Foreign filesystem mountpoints
  - `/media` and `/mnt`

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Important locations in the Red Hat Enterprise Linux Filesystem

Every user on a Red Hat Enterprise Linux system has a *home directory*. This is a directory owned by the user, over which she has complete control. All of the user's personal files go here, as well as any user-specific configuration files In some cases, entire applications can be installed here if they are only for use by that user When a user logs in, she begins in her home directory Root's home directory is `/root`. Most non-root home directories are in the `/home` tree, usually named after the user. So the home directory for user `jane` would be `/home/jane/`

Programs are often referred to as *executables* or *binaries* on a Red Hat Enterprise Linux system. Some binaries are meant for day-to-day use by all users while others, called *system binaries*, are used for system administration and are often restricted to use by the root user The essential binaries necessary to boot and maintain the system reside in `/bin` for regular binaries and `/sbin` for system binaries Non-essential binaries, such as graphical environments, web browsers, office tools and so forth, are installed in `/usr/bin` and `/usr/sbin` The reason for this split is to minimize the size of the root partition On a newly installed system there will also be `/usr/local/bin` and `/usr/local/sbin` directories, but they will be empty Third-party software installed by the administrator, such as software compiled from source code, will usually go in these directories This makes it easier to back up software that is not part of Red Hat Enterprise Linux when wiping and re-installing a system.

When removable media, such as a cdrom or floppy disk, is loaded the filesystem on the media is *mounted* into a subdirectory of `/media`. For example, a cdrom would usually be mounted under `/media/cdrom` and you would access that directory whenever you wanted to read a file from the cdrom. Before ejecting the cd.

`/media/cdrom` would have to be unmounted Filesystems that are on non-removable media but are not part of the Red Hat Enterprise Linux hierarchy are usually mounted under `/mnt`. For example, if your

Copyright © 2006 Red Hat, Inc
All rights reserved.

system dual booted with another operating system like Fedora Core, then the other OS's partitions could be mounted under `/mnt/fedora/`

To learn more about the filesystem hierarchy, visit the Filesystem Hierarchy Standard web site at http://www pathname com/fhs

Copyright © 2006 Red Hat, Inc.
All rights reserved

Other Important Directories

- `/etc` holds system config files
- `/tmp` holds temporary files
- `/boot` holds the kernel and bootloader
- `/var` and `/srv` hold server data
- `/proc` and `/sys` hold system information
- The lib directories hold shared libraries
  - `/lib`, `/usr/lib`, `/usr/local/lib`

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Important Locations (continued)

The previous slide discussed some of the directories most immediately relevant to the Red Hat Enterprise Linux user However, there are a number of other directories worth noting, especially as one moves toward administrating the system.

It is important to remember that on a Red Hat Enterprise Linux system almost everything is configured by variables set within plain-text configuration files Most of these files are stored in the `/etc` directory and its subdirectories In general only root may modify these files. Users can often override system defaults with configuration files stored in their home directories.

The `/tmp` directory is world-writable. In other words, any user can add files to this directory It is usually used by applications for storing temporary data. Once a day the system automatically deletes any files over seven days old in `/tmp` and its subdirectories.

The first component of Red Hat Enterprise Linux to be loaded at boot time is a special program called a *boot loader*. The boot loader is in charge of loading the core of Red Hat Enterprise Linux, called the kernel, into memory. The boot loader, kernel and supporting files such as the boot loader's configuration files, are stored in `/boot`.

The `/var` directory contains regularly-changing system files such as logs, print spools and email spools. It is also used for files being made available by services For example, if you run a web server the html files being made available will usually reside in a subdirectory of `/var/www`. In the future, server data such as this may be moved to the `/srv` directory so that `/var` only contains logs, spools and so forth.

One very valuable resource for learning more about a Red Hat Enterprise Linux system is the `/proc` directory. The files in this directory do not take up space on any disk. Instead they are generated in RAM and updated in real-time to represent information about processes, hardware properties and kernel settings.

Copyright © 2006 Red Hat, Inc.
All rights reserved

The /sys directory contains similar information Using /proc to investigate processes will be discussed in detail later in this class.

The lib directories contain libraries that provide shared code used by many Linux applications. These libraries make developing and updating software easier. The three lib directories are similar to the three usr directories

Copyright © 2006 Red Hat, Inc.
All rights reserved

Current Working Directory

- Each shell and system process has a *current working directory*(cwd)
- **pwd**
  - Displays the absolute path to the shell's cwd

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Current working directory

The current working directory is the directory in which you are currently working. When you type pwd at the command line, the absolute path to your current working directory is displayed. For example:

```
[student@stationX games]$ pwd
/usr/local/games
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

File and Directory Names

- Names may be up to 255 characters
- All characters are valid, except the forward-slash
  - It may be unwise to use certain special characters in file or directory names
  - Some characters should be protected with quotes when referencing them
- Names are case-sensitive
  - Example: `MAIL`, `Mail`, `mail`, and `mAiL`
  - Again, possible, but may not be wise

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Filenames

By default, file names may be up to 255 characters (different restrictions may apply, depending on the particular configuration of your system)

File names generally consist of letters of the alphabet, numbers, and certain punctuation marks. All other characters, except the / character, are valid, but it is often unwise to use certain special characters in file names. Among the characters to avoid are: `> < ? * "` and quotation marks, as well as spaces, tabs and other non-printable characters

To access a file whose name contains special characters, enclose the filename in quotes. For example:

```
[student@stationX ~]$ ls -l "file name with spaces.txt"
-rw-rw-r--  1 student student 0 Dec 14 21:48 file name with spaces.txt
```

Absent the quotes, you would be asking the system to list four different files

File names are case-sensitive. This means that `FILE` is different from `file` and `File`. Again, although it is possible to create these files, it may be unwise to do so, as it may confuse you later

Absolute Pathnames

- Absolute pathnames begin with a forward slash
- Complete "road map" to file location
- Can be used anytime you wish to specify a file name

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Using Absolute Pathnames

The location of a directory or file can be specified by either of two methods: by its *absolute* pathname or its *relative* pathname.

An absolute pathname begins with a slash (/). It contains the name of each directory that must be traversed from the root file system, in order, to reach the object being named, for example:

```
/usr/share/doc/HTML/index.html
```

The file we are referencing, `index.html`, is contained within a directory named `HTML`, which is in turn contained in a directory named doc, which is contained in the directory share, which is contained in directory usr, which is contained in the root (/) directory.

The absolute pathname specifies a 'road map' from the root of the file tree to its location in the file system. This 'road map' is valid regardless of the current directory

Copyright © 2006 Red Hat, Inc.
All rights reserved

Relative Pathnames

- Relative pathnames do not begin with a slash
- Specifies location relative to your current working directory
- Can be used as a shorter way to specify a file name

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Using relative pathnames

A relative pathname does not begin with a slash. It contains the name of each directory that must be traversed from the current directory to reach the object being named. The first component of the pathname *must* exist in the current directory for the pathname to the object to be valid. A filename by itself is a relative pathname; that is, the file must be in the current directory for its reference to be valid.

The special directory name `..` refers to the parent of the current directory, and can be used as part of a pathname.

Some examples of relative pathnames, relative to particular directories, follow. In each case, the file being referenced is `/usr/share/doc/HTML/index.html`.

Current Directory	Relative Path to `index.html`
/usr/share/doc/HTML/	index.html
/usr/share/doc/	HTML/index.html
/usr/share/	doc/HTML/index.html
/usr/	share/doc/HTML/index.html
/	usr/share/doc/HTML/index.html
/usr/share/doc/HTML/en/	../index.html
/usr/share/doc/nautilus-2.1.91/	../HTML/index.html

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Changing Directories

- **cd** changes directories
  - To an absolute or relative path:
    - **cd** `/home/joshua/work`
    - **cd** `project/docs`
  - To a directory one level up:
    - **cd** ..
  - To your home directory:
    - **cd**
  - To your previous working directory:
    - **cd** -

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Getting around with ***cd***

The shell prompt displays the last component of the directory name, for example, `student` to represent `/home/student`. To move from directory to directory on the system, use cd.

The only argument to the **cd** command is either an absolute or relative pathname, or a shortcut representing the directory to which you wish to change

cd .. will switch you to the parent of your current directory.

cd can also be used with no argument to move to your home directory.

The tilde (~) is an abbreviation for 'home directory' Used by itself, it represents your own home directory Used as a prefix to another user's login ID, it represents that user's home directory.

A dash (-) represents your previous working directory. It's a handy shortcut to use to switch back and forth between two directories.

Listing Directory Contents

- Lists the contents of the current directory or a specified directory
- Usage:
  - **ls [options] [files_or_dirs]**

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

*Listing directory contents with **ls***

ls without arguments lists the file and directory names in the current directory

```
[student@stationX ~]$ ls
work
```

ls -a includes so-called "hidden" files and directories whose names begin with a dot:

```
[student@stationX ~]$ ls -a
.bash_history          .bash_logout     .bash_profile     work
```

ls lists another file or directory if given as an argument:

```
[student@stationX ~]$ ls /
bin   dev   home   lib   misc   opt    root   tftpboot usr
boot  etc   initrd  lost+found  mnt   proc  sbin   tmp   var
```

Use ls -l for a more detailed "long" listing:

```
[student@stationX ~]$ ls -l /usr
total 204
drwxr-xr-x    2 root    root    61440 Jul 29 10:07    bin
drwxr-xr-x    2 root    root    4096 Feb  6  1996     dict
drwxr-xr-x    3 root    root    4096 Jul 29 09:00     doc
drwxr-xr-x    2 root    root    4096 Feb  6  1996     etc
...output truncated...
```

ls -R recurses through subdirectories, listing their contents too. **ls -d** lists directory names, not their contents. It has no effect when filenames are passed as arguments. This option is also useful with **-l**:

```
[student@stationX ~]$ ls -ld /usr
drwxr-xr-x    18 root    root    4096 Feb 24 11:36     /usr
```

The **ls** command has many other options. And all options can be used in combination with other **ls** options.

Copying Files and Directories

- **cp** - copy files and directories
- Usage:
  - **cp [options]** ***file destination***
- More than one file may be copied at a time if the destination is a directory:
  - **cp [options]** ***file1 file2 dest***

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copying files with ***cp***

cp must always be given least two arguments When two arguments are given:

The first argument is interpreted as the source file Either an absolute or relative pathname is acceptable.

The second argument is interpreted as the destination Again, use either a relative or absolute pathname. If it names an existing directory, a copy of the source file is placed in that directory with the same name as the source. Otherwise, the destination is interpreted as a file name, and a copy of the source file is created with that destination name

When more than two arguments are given, all arguments but the last are interpreted as source files. The last argument is interpreted as a destination directory Copies of the source files are placed, with their original file names, in the destination directory.

A few common options include:

-i	(interactive): ask before overwriting a file
-r	(recursive): recursively copy an entire directory tree
-p	(preserve): preserve permissions, ownership, and time stamps

Example:

```
[student@stationX ~]$ ls /home/student
testfile
[student@stationX ~]$ cp ~student/testfile /tmp/student_test_file
[student@stationX ~]$ ls /tmp

student_test_file
```

Copying Files and Directories: The Destination

- If the destination is a directory, the copy is placed there with the same name
- If the destination is a file, the copy overwrites the destination
- If the destination does not exist, the copy is created with that name

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The destination affects ***cp****'s behavior*

When copying a single file to a destination, cp first checks to see if a directory exists with the destination name. If it does, a copy of the source file is placed there with its original name. If not, the destination is assumed to be a new file name, and a copy of the source file is made with the destination name.

To illustrate, suppose my current directory is /tmp, and I want to make a copy of file3.txt in a subdirectory of my home directory named backups

```
[student@stationX ~]$ ls -l file3.txt
-rw-rw-r--   1 student student   2633 Feb 22 14:58 file3.txt
[student@stationX ~]$ cp file3.txt ~/backups
[student@stationX ~]$ ls -l ~/backups
-rw-rw-r--   1 student student   2633 Feb 22 14:58 /home/student/backups
```

I've created a file in /home/student called backups, instead of doing what I intended.

The destination directory is actually called backup but I specified backups incorrectly. To ensure the copy is done the way I intend, I'll append the destination directory name with a slash. This indicates that it is absolutely my intention that the destination is a directory, and that a copy of the source file should be placed there. If the destination directory does not exist, the slash will cause the command to fail with an error message.

```
[student@stationX ~]$ cp file3.txt /home/student/backup/
[student@stationX ~]$ ls -l /home/student/backup/
-rw-rw-r--  1 student student   2633 Feb 22 15:03 file3.txt
[student@stationX ~]$ cp file3.txt /home/student/copies/
cp: cannot create regular file `/home/student/copies/file3.txt': ↩
No such file or directory
```

Copyright © 2006 Red Hat, Inc
All rights reserved

Moving and Renaming Files and Directories

- **mv** - move and/or rename files and directories
- Usage:
  - **mv [options] file destination**
  - More than one file may be moved at a time if the destination is a directory
  - **mv [options] file1 file2 dest**

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

*Moving files with **mv***

mv must always be given at least two arguments Aside from a couple of switches, **mv** and **cp** function identically -- the only difference is that **cp** results in matching identical files; with **mv**, the source disappears, leaving only the destination file(s).

When two arguments are given:

The first argument is interpreted as the source file. It may or may not be prepended with a relative or absolute path name.

The second argument is interpreted as the destination It may or may not be prepended with a relative or absolute path name If it names an existing directory, the source file is moved to that directory with the same name as the source. Otherwise, the destination is interpreted as a file name, and the source file is moved and/or renamed to that destination name

When more than two arguments are given:

All arguments but the last are interpreted as source files They may or may not be prepended with a relative or absolute path name

The last argument is interpreted as a destination directory. It may or may not be prepended with a relative or absolute path name. The source files are moved, with their original file names, to the destination directory.

Example:

```
[student@stationX ~]$ ls  ~student
testfile
[student@stationX ~]$ mv ~student/testfile /tmp/student_test_file
[student@stationX ~]$ ls  ~student
[student@stationX ~]$ ls /tmp/
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

```
student_test_file
....other output omitted...
```

Copyright © 2006 Red Hat Inc
All rights reserved

Moving and Renaming Files and Directories: The Destination

- If the destination is a directory, the source is moved there with the same name
- If the destination is a file, the source overwrites the destination
- If the destination does not exist, the source is renamed

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*The destination affects **mv**'s behavior*

When moving a single file to a destination, mv first checks to see if a directory exists with the destination name If so, the source file is moved to that directory If not, the destination is assumed to be a new file name, and the source file is renamed to that destination name. Like cp, to ensure that the destination is interpreted as a directory name and not a file name, append it with a slash:

```
[student@stationX ~]$ mv procedure.txt.bak2 /home/student/procedures/
```

If the destination directory does not exist, the slash will cause the command to fail with an error message.

To rename a directory, or to move a directory and all its contents to another location, just use the mv command as if you were moving files If the destination directory exists, the source directory is moved into that destination directory, with the same name, as a new subdirectory.

If the destination directory does not exist (but the destination pathname is valid), the source directory will be moved to the destination directory with the new name.

```
[student@stationX ~]$ ls -l
total 4    drwxr-x---    2 student  student  4096 Sep  9 18:16   foo
[student@stationX ~]$ mv foo bar
[student@stationX ~]$ ls -l
total 4    drwxr-x---    2 student  student  4096 Sep  9 18:16   bar
```

Creating and Removing Files

- **rm** - remove files
- **rm [options] <filename>....**
  - **-i** (interactive)
  - **-r** (recursive)
  - **-f** (force)
- **touch** - create empty files or update file timestamps

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Removing files with ***rm***

rm removes files. One or more files can be targets for removal. By default, **rm** will not remove directories. The **-r** option tells **rm** to remove files recursively and thus it will delete directories and their contents.

To remove a directory and all files and subdirectories it contains, suppressing and ignoring warnings about removing write-protected files and directories, use **rm -rf**. Be very careful not to erase needed files with this option! If necessary, **rm** command can be made interactive with **-i**

There is no way to undo the effects of **rm**, except to restore from a backup.

Examples:

Remove a directory and all its contents

```
[student@stationX ~]$ rm -r include
```

Use **-r** in combination with **-i** to recurse through the named directory and to query whether or not each file should be removed. The file will be removed if the user types **y** or **Y**.

```
[student@stationX ~]$ rm -ri include
rm: descend into directory `include'? y
rm: descend into directory `include/readline'? y
rm: remove regular file `include/readline/readline.h'? y
....output truncated...
```

Use **-f** to suppress warnings about write-protected files. In the example above had used **rm -rf** instead of **rm -ri**, the directory tree would have been deleted with no further prompting.

Creating and modifying files with ***touch***

Copyright © 2006 Red Hat Inc.
All rights reserved

touch updates a file's timestamps. For example, if the last time you accessed a file was at 10:02 pm and you touch the file at 10:45 pm, the file will show its last access at 10:45 pm. If you touch a file that does not exist, an empty file will be created

Copyright © 2006 Red Hat, Inc.
All rights reserved

Creating and Removing Directories

- **mkdir** creates directories
- **rmdir** removes empty directories
- **rm -r** recursively removes directory trees

17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Directory management

Directories are created with **mkdir**. For example, to create a directory called work:

```
[student@stationX ~]$ mkdir work
```

To remove an empty directory, use **rmdir**. For example:

```
[student@stationX ~]$ rmdir work
```

rmdir will only remove empty directories. To remove a directory and its contents, use **rm -r**.

Copyright © 2006 Red Hat Inc
All rights reserved

Using Nautilus

- Gnome graphical filesystem browser
- Can run in "Spatial or Browser" mode
- Accessed via.....
  - Desktop icons
    - Home: Your home directory
    - Computer: Root filesystem, network resources and removable media
  - "File Browser" option on Applications menu

18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat. com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Browsing the filesystem graphically

Nautilus is a graphical filesystem browser provided as part of the Gnome graphical environment. While managing files from the command line can be more versatile and powerful, many people consider a graphical interface more intuitive Nautilus can run in one of two modes, each of which has a slightly different interface.

"Spatial" mode is designed to be the most intuitive for new users and the simplest in terms of user interface clutter. Windows have a very basic layout with no toolbar and when a directory is double-clicked it opens in its own window. A menu in the lower-left of each window allows the user to list and select parent directories of the one being currently displayed Typing *Ctrl-Shift-w* closes all parent windows

"Browser" mode is a more tradition file manager interface with a side pane on the left that can display file details, a filesystem tree, or even notes the user has taken about the contents of a directory What is displayed in the sidebar can be selected from a dropdown menu. Folders open in the same window instead of creating new ones and the user navigates using the standard **back, forward** and **up** icons available in most graphical file managers.

Nautilus can be accessed in a number of ways. The two most common are via desktop icons or a menu option. The desktop icons labeled **Computer** and, for example, **Joe's Home** each open a Nautilus window in spatial mode The **Home** icon opens a window that displays the user's home directory. The **Computer** icon opens a window that presents the user with an ordered view of available filesystem resources. This includes the **Filesystem** icon, which opens a window displaying the root directory, the Network icon, which allows the user to browse the local network for shared disks, and an icon for each removable-media filesystem, such as those on cdroms, floppy disks and USB drives.

Nautilus can be started in browser mode by selecting File Browser from the Applications menu in the upper-right corner of your Gnome desktop. If you would like Nautilus to always start in browser mode,

Copyright © 2006 Red Hat Inc
All rights reserved.

even when accessed via the desktop icons, open a Nautilus window, select Edit->Preferences from the menus, go to the **Behavior** tab, check **Always open in browser windows** and click **Ok**.

Copyright © 2006 Red Hat Inc
All rights reserved.

Moving and Copying in Nautilus

- Drag-and-Drop
  - Left-button: Move on same filesystem, copy on different filesystem
  - Ctrl-Left-button: Always copy
  - Alt-Left-button: Ask whether to copy, move or create symbolic link (alias)
- Context menu
  - Right-click to rename, cut, copy or paste

19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Managing files graphically

To drag and drop files, you would normally use the left-button If the destination is on the same filesystem as the source, it will move the file/directory. If the destination is on a different filesystem, it will copy the file/directory. To force it to always copy, hold down the Ctrl key while dragging with the left button. To have nautilus ask whether to copy, move or create a symlink (an alias to the source file that doesn t take up extra space on the drive), hold down the Alt key while dragging.

If you click on an object in Nautilus with the right mouse button you will be presented with a "context menu" presenting actions you may wish to take on that item. The exact contents of the context menu will change depending on what is being clicked (the "context" of the click). For files or directories the context menu will include options to copy, cut (move) or rename the selected object If you cut or copy a file or directory and then right-click on your desktop or a directory then the context menu will have a paste option. Selecting this option will copy or move (depending on whether the source file was copied or cut from its original location) the previously selected object (or objects) into the directory.

Copyright © 2006 Red Hat Inc
All rights reserved.

Determining File Content

- Files can contain many types of data
- Check file type with file before opening to determine appropriate command or application to use
- **file [options] <filename>...**

20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

*Using the **file** command*

The contents of any given file might be ASCII (plain text, HTML, executable shell scripts, C program source code, mailbox-format text) or binary (compiled executables, compressed data, images, and sound samples).

Binary file types use an extended character set. Some of these characters are also used to display special characters, sound an error beep at the terminal, clear or flash the screen, or even lock the terminal display. Before displaying the contents of a file in a terminal window, it is often wise to check its file type with file. The file type reported will help determine the appropriate command or application to use to access the file.

file prints its best guess of the type of data contained in a file whose name is given as an argument. It bases its guess on a comparison of the contents of the file and the patterns and offsets in its reference file, `/usr/share/magic`. Some file types, as reported by **file**:

File Name	File Type
`bookmarks.html`	HTML document text
`carrental.ps`	PostScript document text conforming at level 3.0
`procmailrc.nospam`	ASCII English text
`snifob`	perl script text executable
`xfonts.txt`	ASCII mail text
`Girl_Next_Door.mp3`	MP3, 128 kBits, 44.1 kHz, JStereo
`pan-0.10.0.91.tar.bz2`	bzip2 compressed data, block size = 900k
`pic2.jpg`	JPEG image data, JFIF standard 1.01, resolution (DPI), "File written by Adobe Photoshop", 72 x 72
`rpmfind-1.7-1.i386.rpm`	RPM v3 bin i386 rpmfind-1.7-1

Copyright © 2006 Red Hat, Inc.
All rights reserved

`xsel`	ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), stripped

Copyright © 2006 Red Hat Inc
All rights reserved.

Viewing an Entire Text File

- Syntax:
  - **cat [options] [<file>...]**
- Contents of the files are displayed sequentially with no break
- Files display "con**cat**enated"

21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Dumping files with cat

cat opens the files given as its arguments and displays their contents to the terminal.

To see how **cat** works, try running **cat /etc/profile**

Unless you can read very fast, you probably noticed a problem with the output scrolling off the top of the page too quickly. **cat** is most useful for viewing short files; other commands, such as less, are more suited to viewing larger files.

Some useful options to use with **cat**:

-A	Show all characters, including control characters and non-printing characters
-s	"Squeeze" multiple adjacent blank lines into a single blank line
-b	Number each (non-blank) line of output

Copyright © 2006 Red Hat Inc
All rights reserved.

Viewing Text Page by Page

- **less [options] [filename]**
- Scroll with arrows/*PgUp*/*PgDn*
- Useful commands while viewing:
  - `/text` searches for *text*
  - `n`/`N` jumps to the next/previous match
  - `v` opens the file in a text editor
- **less** is the pager used by **man**

22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Navigating text with ***less***

Space	moves ahead one full screen
`b`	moves back one full screen
Enter	moves ahead one line
k	moves back one line
`g`	moves to the top of the file
`G`	moves to the bottom of the file
`/text`	searches for text
`n`	repeats the last search
`N`	repeats last search, but in the opposite direction
`q`	quits
`v`	opens the file in a text editor (**vi** by default)

End of Unit 3

- Questions and Answers
- Summary
  - The Linux filesystem hierarchy
  - Command-line file management tools
  - The Nautilus file manager

23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat Inc
All rights reserved.

Lab 3
Browsing the Filesystem

Goal: Become familiar with the functions, syntax, and use of several essential file and directory manipulation commands.

Estimated Duration: 30 minutes

System Setup: A working, installed Red Hat Linux system with an unprivileged user account named student with a password of student.

Copyright © 2006 Red Hat Inc
All rights reserved

Sequence 1: Directory and File Organization

Scenario: A few files have accumulated in your home directory, and you have decided that it is time to organize things. You plan to create several new subdirectories, and to copy and move your files around to fit into your new scheme. Additionally, you have several files that are not needed at all, which must be deleted.

Deliverable: A more organized home directory, with files placed into the appropriate subdirectories.

Instructions:

1. Log in on `tty1` (*Ctrl-Alt-F1*) as user `student` with the password of *`student`*.

2. Immediately after logging into the system, you should be in your home directory. Verify this using the "print working directory" command.

   ```
   [student@stationX ~]$ pwd
   /home/student
   ```

3. Check to see if you have any files in your home directory using each of the following commands:

   ls
   ls -a
   ls -al

   Why do the first and second command return different numbers of files?

   What is the size of the largest file and the largest directory currently in your home directory as reported by the third command?

4. You will now use `touch` to create the files needed for this sequence. The details of how the expansion used in the following command works will be covered in a later Unit. For now, simply type the following line exactly as you see it (with the curly braces {} included, and an underscore character between the first two groups of sets):

   ```
   [student@stationX ~]$ touch {report,graph}_{jan,feb,mar}
   ```

Copyright © 2006 Red Hat Inc
All rights reserved

5. Use the ls command to examine the results of the last command. You should find that it created the following six new, empty files in your home directory.

```
[student@stationX ~]$ ls
graph_feb  graph_jan  graph_mar  report_feb  report_jan report_
```

These files represent the data files that you will use in the remainder of this sequence. If for some reason you do not see these files, ask the instructor for assistance; without these files, the remainder of this lab will not work.

6. In order to organize your files you must first create some new directories. Use **mkdir** to create a few directories. As you change directories, below, be sure to check that your working directory is as expected.

```
[student@stationX ~]$ mkdir Projects
[student@stationX ~]$ mkdir Projects/graphs
[student@stationX ~]$ cd Projects
[student@stationX Projects]$ mkdir reports
[student@stationX Projects]$ cd reports
[student@stationX reports]$ mkdir ../Backups
```

Use **ls** to examine your work:

```
[student@stationX reports]$ cd
[student@stationX ~]$ ls -l
total 4
-rw-rw-r--  1 student  student     0 Sep 30 21:08 graph_feb
-rw-rw-r--  1 student  student     0 Sep 30 21:08 graph_jan
-rw-rw-r--  1 student  student     0 Sep 30 21:08 graph_mar
drwxrwsr-x  5 student  student  4096 Sep 30 21:09 Projects
-rw-rw-r--  1 student  student     0 Sep 30 21:08 report_feb
-rw-rw-r--  1 student  student     0 Sep 30 21:08 report_jan
-rw-rw-r--  1 student  student     0 Sep 30 21:08 report_mar
[student@stationX ~]$ ls Projects
Backups  graphs  reports
```

Copyright © 2006 Red Hat, Inc
All rights reserved.

7 Begin by moving all of the graph files into the graphs subdirectory of the Projects directory Do this in two steps: in the first step, move one file; in the second step, move two files:

```
[student@stationX ~]$ mv graph_jan Projects/graphs
[student@stationX ~]$ mv graph_feb graph_mar Projects/graphs
[student@stationX ~]$ ls -l Projects/graphs/
total 0
-rw-rw-r--  1 student   student      0 Sep 30 21:08 graph_feb
-rw-rw-r--  1 student   student      0 Sep 30 21:08 graph_jan
-rw-rw-r--  1 student   student      0 Sep 30 21:08 graph_mar
```

8. Next, move two of the "report" files into the reports subdirectory of the Projects directory. Move the files in one command:

```
[student@stationX ~]$ mv report_jan report_feb Projects/reports
[student@stationX ~]$ ls -l Projects/reports
total 0
-rw-rw-r--   1 student   student      0 Sep 30 21:08 report_feb
-rw-rw-r--   1 student   student      0 Sep 30 21:08 report_jan
```

9. Remove the remaining report file:

```
[student@stationX ~]$ rm report_mar
[student@stationX ~]$ ls
Projects
```

10. Change into the Backups directory and copy the January files into this directory. Copy one using an absolute pathname and the other using a relative pathname:

```
[student@stationX ~]$ cd Projects/Backups
[student@stationX Backups] pwd /home/student/Projects/Backups
[student@stationX Backups] cp ../reports/report_jan .
[student@stationX Backups] cp ↙
/home/student/Projects/graphs/graph_jan .
[student@stationX Backups] ls -l
total 0
-rw-rw-r--   1 student   student      0 Sep 30 21:20 graph_jan
-rw-rw-r--   1 student   student      0 Sep 30 21:20 report_jan
```

The trailing dot is the destination: the present working directory.

Copyright © 2006 Red Hat Inc.
All rights reserved

Sequence 2: Managing Files with Nautilus

Scenario: A co-worker, Bob, would like to see a copy of your graphs. You will take advantage of the /tmp directory to make these files available to him using Nautilus. The instructions below use Nautilus' "Spatial" mode and the copy/paste method for copying files. Nautilus is a very flexible tool and there are multiple ways to approach most problems. If you have time left over after completing this sequence, try discovering some of these other methods. For example, you can try doing the sequence again using Nautilus' "Browser" interface to see which you prefer. To launch the Nautilus browser, select Filesystem Browser from the Applications menu. Be sure to take advantage of the filesystem tree feature if you do this (select Tree from the dropdown in the left sidebar).

Deliverable: Bob can now access copies of your graphs by retrieving them from `/tmp/Stuff_for_Bob`.

Instructions:

1. Double-click on the **student's Home** icon on your desktop. This will open a Nautilus window displaying your home directory.

2. Double-click on **Projects**, then **graphs**. Note that each directory opens in a new window.

3. With the **graphs** window selected, press *Ctrl-Shift-w* to close the parent directory windows.

4. Press *Ctrl-a* to select all three graphs.

5. Press *Ctrl-c* to copy the files. A message will be displayed at the bottom of the window informing you that three items have been copied.

6. Press *Ctrl-l* (the letter l, not the number 1) to display the Open Location dialog. Type **`/tmp`** (note that you can use tab-completion) and press Enter to open a new window displaying `/tmp`.

7. Press *Ctrl-Shift-n* or right-click on the window background and select Create Folder to create a new directory.

8. Type in **`Stuff_for_Bob`** as the new directory's name and double-click the directory to open it

9. Press *Ctrl-v* to paste your graphs into `/tmp/Stuff_for_Bob`.

Copyright © 2006 Red Hat Inc.
All rights reserved.

10 Press *Ctrl-q* to close all Nautilus windows

Copyright © 2006 Red Hat Inc
All rights reserved

Sequence 1 Solutions

3 **ls** returns fewer files than **ls -a** because the **-a** option includes files whose names begin with a period Such files are usually used for storing configuration information and are not included in directory listings by default.

The fifth column of **ls -l**'s output displays the file's size.

Copyright © 2006 Red Hat Inc
All rights reserved.

Unit 4

The bash Shell

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Objectives

Upon completion of this unit, you should:

- Know how to use command-line shortcuts
- Understand command-line expansion
- Be able to inhibit command-line expansion
- Know how to use history and editing tricks

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

bash Introduction

- "Bourne Again Shell"
- Successor to **sh**, the original Unix shell
- Developed for the GNU Project
- The de facto standard Linux shell
- Backward-compatible with Bourne shell (**sh**) - the original (standard) UNIX shell.

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

bash

The name "Bourne Again Shell" is consistent with many of the humorous naming conventions adopted by the GNU project. "GNU", for instance, is a recursive acronym that stands for "GNU's Not Unix"

Copyright © 2006 Red Hat, Inc
All rights reserved.

bash Heritage and Features

- Bourne Again Shell **(bash)**
  - Implements many of the best features from earlier shells: **sh, csh, ksh, tcsh**
    - Command line completion
    - Command line editing
    - Command line history
    - Sophisticated prompt control

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat Inc
All rights reserved

Command Line Shortcuts
File Globbing

- Globbing is wildcard expansion:
  - `*` - matches zero or more characters
  - `?` - matches any single character
  - `[a-z]` - matches a range of characters
  - `[^a-z]` - matches all except the range

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Globbing

When typing commands, it is often necessary to issue the same command on more than one file at the same time. The use of wildcards, or metacharacters, allows one pattern to expand to multiple filenames by a process called globbing. For example, if a directory contains the following files: `joshua.txt James.txt alex.txt Angelo.txt gonk.mp3 zonk.mp3`

Typing the following command:

```
[student@stationX ~]$ rm *mp3
```

is the same as typing:

```
[student@stationX ~]$ rm gonk.mp3 zonk.mp3
```

The result is that all files in the directory that have names ending in mp3 (in this case, just the two listed) will be removed

In addition to wildcards, ranges of characters can be specified within straight-braces (?[? and ?]?) Red Hat Enterprise Linux uses UTF-8 encoding of characters, which means that each capital letter comes directly after the corresponding lower-case letter. As a result, a command like

```
[student@stationX ~]$ ls [a-j]*.txt
```

would list `alex.txt`, `Angelo.txt` and `joshua.txt` but not `James.txt`.

echo can be used to test the expansion of metacharacters before using them in a destructive command like **rm**:

```
[student@stationX ~]$ echo ?o*
joshua.txt    gonk.mp3  zonk.mp3
```

Copyright © 2006 Red Hat Inc
All rights reserved

Command Line Shortcuts
The *Tab* Key

- Type *Tab* to complete command lines:
  - For the command name, it will complete a command name
  - For an argument, it will complete a file name
- Examples:

```
$ xte<Tab>
$ xterm
$ ls myf<Tab>
$ ls myfile.txt
```

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

File and command completion

The tab key can be used to complete either a command name or a file name. To complete a file name, type in the command and begin typing a file name. When you think you have typed in a unique set of characters, hit the tab key. If you succeeded in issuing a unique set of characters, the tab key will complete the file name for you. For example:

```
[student@stationX ~]$ ls myf<Tab>
```

If the only file in the present working directory beginning with `myf` is `myfile.txt`, then this file name will appear on your screen:

```
[student@stationX ~]$ ls myfile.txt
```

In both cases, if there are multiple possible matches, a second tab will list the possible matches. For example:

```
[student@stationX ~]$ ls
dove  eagle  pelican  penguin
[student@stationX ~]$ cat p<Tab>
pelican  penguin
[student@stationX ~]$ cat pe
```

Note that the tab key added the e, which is common to both choices. You may then add an extra character to specify the particular item that you want listed. The tab key can also be used to complete command

Copyright © 2006 Red Hat Inc
All rights reserved.

names, using the same procedure Try issuing two <*Tab*> keys on a command line before typing any characters

Command Line Shortcuts
History

- **bash** stores a history of commands you've entered, which can be used to repeat commands
- Use **history** command to see list of "remembered" commands

```
$ history
14  cd /tmp
15  ls -l
16  cd
17  cp /etc/passwd .
18  vi passwd
.... output truncated ...
```

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

History

In addition to the basic command recall with the arrow keys, the bash history mechanism supports a variety of advanced ways of retrieving commands from the history list.

!!	repeats last command
!*char*	repeats last command that started with *char*
!*num*	repeats a command by its number in history output

To view past commands with the history numbers, use the **history** command.

Other slightly more advanced history tricks include:

!?abc	repeats last command that contains (as opposed to ?started with?) abc
!-n	repeats a command entered n commands back

Use **^old^new** to repeat the last command with old changed to new, for example:

```
[student@stationX ~]$ cp filter.c /usr/local/src/project
[student@stationX ~]$ ^filter^frontend
cp frontend.c /usr/local/src/project
```

Copyright © 2006 Red Hat Inc
All rights reserved

More History Tricks

- Use the *up* and *down* keys to scroll through previous commands
- Type *Ctrl-r* to search for a command in command history.
  - *(reverse-i-search)`':*
- To recall last argument from previous command:
  - *Esc,* . (the escape key followed by a period)
  - *Alt-* . (hold down the alt key while pressing the period)

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

History

Using your history is a great productivity-enhancing tool. Linux users who develop a habit of using their history can streamline and speed their use of the shell. Try playing with the keystrokes listed above

You can ignore repeated duplicate commands and repeated lines that only differ in prepended spaces by adding the following to your `.bashrc`

```
[student@stationX ~] export HISTCONTROL=ignoreboth
```

Copyright © 2006 Red Hat Inc
All rights reserved

Command Line Expansion
The tilde

- Tilde (~)
- May refer to your home directory

```
$ cat ~/.bash_profile
```

- May refer to another user's home directory

```
$ ls ~julie/public_html
```

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Tilde Expansion

The tilde (~) expansion feature is borrowed from the C Shell and makes it easy to reference files and directories inside your or someone else's home directory. The ~ character, called a tilde, expands to your own home directory, or, if followed by a user name, by that user's home directory.

For example, the string `~/.bash_profile` means ?the `.bash_profile` in my home directory?, whereas `~sally/.bash_profile` means ?the `.bash_profile` in sally's home directory?. An advantage is that you do not need to know sally's home directory to reference a file within it, useful in environments where home directories exist in non-standard locations

Command Line Expansion
Variable and String

- Parameter/Variable: **$**
  - Substitute the value of a variable in a command line

```
$ cd $HOME/public_html
```

- Curly braces: { }
  - A string is created for every pattern inside the braces regardless if any file exists

```
$ rm hello.{c,o}
```

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Words on the command line preceded by a dollar sign are considered to be variables; the shell will replace the string with the value of the variable before calling the appropriate command. For example:

```
[student@stationX ~]$ cd $HOME/public_html
```

Before the cd command is called, the shell will expand $HOME to its appropriate value. In this case, $HOME expands to the user's home directory. Thus, for user sally, the command will expand to:

```
[student@stationX ~]$ cd /home/sally/public_html
```

To see a list of variables and their values, run the **set** command.

Curly braces expand to create string patterns they describe. Commas separate the various string patterns and an initial comma means that the null string is one of the requested patterns. For example:

```
[student@stationX ~]$ echo {a,b}
a b
[student@stationX ~]$ echo x{a,b}
xa xb
[student@stationX ~]$ cp file.txt{,-save}
cp file.txt file.txt-save
```

Curly braces are useful for generating patterned strings. For example:

```
[student@stationX ~]$ mkdir -p work/{inbox,outbox,pending}/{normal,urgent,imp
[student@stationX ~]$ ls work
inbox  outbox  pending
```

Copyright © 2006 Red Hat Inc
All rights reserved

```
[student@stationX ~]$ ls work/inbox
important  normal  urgent
```

Without the curly braces, the **mkdir** command above would take almost two hundred keystrokes to execute.

Copyright © 2006 Red Hat Inc
All rights reserved.

Command Line Expansion
Command and Math

- Command Output - ` ` or $ ( )
  - Substitute output from a command in a command line

```
$ echo "Hostname: `hostname`"
```

- Arithmetic - $ [ ]
  - Substitute result of arithmetic expression in a command line

```
$ echo Area: $[ $X * $Y ]
```

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat .com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Command substitution

Use of the backquotes is called command substitution. In command substitution, the command in backquotes is executed and the output of the command is placed on the command line as if the user had typed it in An alternative syntax for the backquotes is $() The example in the slide above could have been written as:

```
[student@stationX ~]$ echo "Hostname: $(hostname)"
```

The traditional (Bourne shell) method of doing arithmetic at the shell command line is to use expr and command output substitution:

```
[student@stationX ~]$ echo Area: `expr $X \* $Y`
```

Arithmetic substitution

The use of the **expr** command requires careful syntax, including putting the backslash before the asterisk and ensuring that each element within the evaluation is a separate shell word An alternative method for performing simple mathematical functions is the use of the **$ [ ]** syntax Using this syntax, it is not necessary to use backslashes before the asterisk, nor does spacing within the square brackets matter, thus making the syntax significantly easier:

```
$ echo Area: $[ $X * $Y ]
```

Basic arithmetic evaluations are recognized:

+	addition

Copyright © 2006 Red Hat Inc
All rights reserved

`-`	subtraction
`*`	multiplication
`/`	division
`**`	exponentiation
`%`	modulo (remainder after division)

More complex evaluations are described in the ARITHMETIC EVALUATIONS section of the bash man page

Copyright © 2006 Red Hat Inc
All rights reserved

Protecting from Expansion
Backslash

- Backslash (\ ) makes the next character literal

```
$ echo Your cost: \$5.00
```

- Used as last character on line to "continue command on next line"

```
$ echo "This long string will be echoed \
> back as one long line"
This long string will be echoed back as one long line
```

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Escaping characters

Protecting characters from expansion with the backslash (\) is very important for some operations, such as when using find to print filenames that match a pattern:

```
[student@stationX ~]$ find / -name foo*
```

In this case the * is expanded by the shell. If the pattern matches a single filename in the current directory (such as foobar or foofram), the expanded name will be passed to find as a parameter Thus, the find command will return only files with that specific name, instead of files with the pattern **foo***

To find all file and directory names that begin with foo, "escape" the wildcard character to pass it intact to **find**.

```
[student@stationX ~]$ find / -name foo\*
```

Copyright © 2006 Red Hat Inc
All rights reserved

Protecting from Expansion
Quotes

- Quoting prevents expansion
  - Single quotes (') inhibit all expansion
  - Double quotes (") inhibit all expansion, except:
    - $ (dollar sign) - variable expansion
    - ` (backquotes) - command substitution
    - \ (backslash) - single character inhibition
    - ! (exclamation point) - history substitution

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Inhibiting expansion with quotes

The use of the backslash to inhibit shell expansion of the command line can be burdensome if many characters need to be escaped. For example:

```
[student@stationX ~]$ echo \*\*\* SALE \*\*\*
```

Not only is this displeasing to the eye, but it also invites typing errors. To inhibit multiple characters from being expanded it is useful to use either single or double quotes:

```
[student@stationX ~]$ echo '*** SALE ***'
```

The difference between using single and double quotes is that single quotes will inhibit just about all command line expansion, whereas double quotes will inhibit some expansion, but allow others As a rule of thumb, double quotes inhibit file name generation expansion, but not other types

In many cases, using either single or double quotes will yield identical results. For example, this will yield the same results as the single quoted example above:

```
[student@stationX ~]$ echo ?*** SALE ***?
```

But consider in the following examples Very different strings can result from different quotes:

```
[student@stationX ~]$ echo "The current date is `date`"
The current date is Sat Apr 27 17:45:25 EDT 2002
[student@stationX ~]$ echo 'The current date is `date`'
The current date is `date`
```

Copyright © 2006 Red Hat Inc
All rights reserved.

Command Editing Tricks

- *Ctrl-a* moves to beginning of line
- *Ctrl-e* moves to end of line
- *Ctrl-u* deletes to beginning of line
- *Ctrl-k* deletes to end of line
- *Ctrl-`arrow`* moves left or right by word

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Command line editing

Combining command line editing with history is great way to easily modify previously-run commands The default key bindings in bash are the same as those in the text editor **emacs**; **vi**-style bindings are also available for use.

Copyright © 2006 Red Hat Inc
All rights reserved

gnome-terminal

- Applications->System Tools->Terminal
- Graphical terminal emulator that supports multiple "tabbed" shells
  - *Ctrl-Shift-t* creates a new tab
  - *Ctrl-PgUp/PgDn* switches to next/prev tab
  - *Ctrl-Shift-c* copies selected text
  - *Ctrl-Shift-v* pastes text to the prompt

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

End of Unit 4

- Questions and Answers
- Summary
  - Command expansion: `$()`
  - Arithmetic expansion: `$[]`
  - History recall: `!`*`string`*, `!`*`num`*
  - Inhibition: `''`, `\`

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat Inc
All rights reserved

Lab 4
Exploring the Bash Shell

Goal:	Become familiar with the functions, syntax, and use of several essential file and directory manipulation commands. Practice combining these commands together in useful ways to accomplish common user tasks.
Estimated Duration:	45 minutes
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.

Sequence 1: Directory and file organization

Scenario: Once again files have managed to accumulate in your home directory and you have decided that it is time to organize things. This time you will use your knowledge of the **bash** shell to perform more complex file management tasks.

Deliverable: A more organized home directory, with files placed into the appropriate subdirectories, and some files backed up to `/tmp/archive`.

Instructions:

1. Log in on `tty1` as user `student` with the password of *`student`*.

2. Immediately after logging into the system, you should be in your home directory. Verify this with **pwd**.

   ```
   [student@stationX ~]$ pwd
   /home/student
   ```

   Note that you can also tell you are in your home directory by noting the ~ in your command prompt.

3. You will now use touch to create the files needed for this sequence. The details of how the expansion used in the following command works will be covered in a later Unit. For now, simply type the following line exactly as you see it (with the curly braces {} included, and an underscore character between the first few groups of sets). Have another nearby student, or the instructor verify the accuracy of your command before you press enter:

   ```
   [student@stationX ~]$ touch ↩
   {report,memo,graph}_{sep,oct,nov,dec}_{a,b,c}_{1,2,3}
   ```

4. Use the **ls** command to examine the results of the last command. You should find that it created 108 new, empty files (you don't need to count) in your home directory. These files represent the data files that you will use in the remainder of this sequence. If for some reason you do not see these files, ask the instructor for assistance; without these files, the remainder of this lab will not work.

Copyright © 2006 Red Hat Inc
All rights reserved.

5. In order to organize your files you must first create some new directories Use mkdir to create some subdirectories directly inside your home directory:

```
[student@stationX ~]$ mkdir a_reports
[student@stationX ~]$ mkdir september october november december
```

Again, use **ls** to examine your work.

6. Create some additional subdirectories inside one of your new directories using the following commands.

```
[student@stationX ~]$ cd a_reports
```

to change to the directory Then:

```
[student@stationX a_reports]$ mkdir one two three
```

Use **ls** to verify that you have three new directories named one, two, and three under your `a_reports` subdirectory.

7. Begin by moving all of the "b" reports out of your home directory and grouping them by month. When working with complicated wildcard patterns, it is a good idea to pre-verify the operation to ensure you are operating on the correct files. One way to do this is to replace your command with a harmless command using the intended wildcard pattern.

```
[student@stationX a_reports]$ cd
[student@stationX ~]$ ls -l *dec_b_?
```

You should see the 9 "december", "b" files listed Move one of them to the december directory:

```
[student@stationX ~]$ mv graph_dec_b_1 december
```

Now move the rest of them them with:

```
[student@stationX ~]$ mv *dec_b_? december
```

List the contents of the december subdirectory to verify the move operation was successful:

```
[student@stationX ~]$ ls -l december
total 9
-rw-rw-r--  1 student  student   0 Oct 16 22:17 graph_dec_b_1
```

Copyright © 2006 Red Hat Inc
All rights reserved.

```
-rw-rw-r--  1 student  student   0 Oct 16 22:16 graph_dec_b_2
-rw-rw-r--  1 student  student   0 Oct 16 22:16 graph_dec_b_3
-rw-rw-r--  1 student  student   0 Oct 16 22:16 memo_dec_b_1
-rw-rw-r--  1 student  student   0 Oct 16 22:16 memo_dec_b_2
-rw-rw-r--  1 student  student   0 Oct 16 22:16 memo_dec_b_3
-rw-rw-r--  1 student  student   0 Oct 16 22:16 report_dec_b_1
-rw-rw-r--  1 student  student   0 Oct 16 22:16 report_dec_b_2
-rw-rw-r--  1 student  student   0 Oct 16 22:16 report_dec_b_3
```

8. Move all of the remaining "b" reports into their respective directories:

```
[student@stationX ~]$ mv *nov_b_? november
[student@stationX ~]$ mv *oct_b_? october
[student@stationX ~]$ mv *sep_b_? september
```

9. You will now collect the "a" reports into their respective corresponding numbered directories. Notice the use of ~ as shorthand for "your home directory". The combination of the wildcard and the pattern specifies all files that end in `_a1` in your home directory.

```
[student@stationX ~]$ cd a_reports
[student@stationX a_reports]$ mv ~/*_a_1 one/
```

The "september" "a1" files are old and no longer needed. Use echo to make sure you've created a pattern that matches only these files, then delete them, and verify that the other "a1" files were moved properly:

```
[student@stationX a_reports]$ cd one
[student@stationX one]$ echo *sep*
[student@stationX one]$ rm *sep*
[student@stationX one]$ ls
graph_dec_a_1  graph_oct_a_1  memo_nov_a_1  report_dec_a_1
report_oct_a_1 graph_nov_a_1  memo_dec_a_1   memo_oct_a_1
report_nov_a_1
```

10. Move the final "a_2" and "a_3" reports into their respective directories. To make life interesting, we'll move them from the current directory, using both relative and absolute pathnames. First, use pwd to identify the current directory:

```
[student@stationX one]$ pwd
/home/student/a_reports/one
```

Verify the pattern that references the "a_2" files with echo, then move them using absolute pathnames:

Copyright © 2006 Red Hat inc
All rights reserved.

```
[student@stationX one]$ echo /home/student/*a_2*
[student@stationX one]$ mv /home/student/*a_2* ↵
/home/student/a_reports/two
```

Even though your current directory is /home/student/a_reports/one, you can move files from /home/student to /home/student/a_reports/two because you specified the files' pathnames - in this case, absolute pathnames

Now move the "a_3" files using relative pathnames. Again, first verify the pattern references the correct files.

```
[student@stationX one]$ echo ../../*a_3*
[student@stationX one]$ mv ../../*a_3* ../three
```

11. Return to your home directory, and use **ls** to verify that the only files remaining in this directory are the "c" files (ie. graph_dec_c_1, graph_dec_c_2, ...)

12. The "c1" and "c2" report files for each month are important, and you want to make a backup of them in another directory:

```
[student@stationX ~]$  mkdir /tmp/archive
[student@stationX ~]$  cp report*[12] /tmp/archive/
```

Additionally, all the report files for the month of December should be backed up to the /tmp/archive directory. Note the use of the **-i** option to have **cp** prompt before overwriting any files.

```
[student@stationX ~]$ cp -i report_dec* /tmp/archive/
cp: overwrite `/tmp/archive/report_dec_c_1'? n
cp: overwrite `/tmp/archive/report_dec_c_2'? n
```

13. Now that you have backed up the few "c" files that are important to you, you want to delete all of the files still remaining in your home directory. Examination of the remaining files reveals that the wildcard ***c*** will match all of them. However, to ensure that you do not accidentally delete other files, try out the following commands, examining the output:

```
[student@stationX ~]$ ls *c*
...output omitted...
[student@stationX ~]$ ls -Fd *c*
...output omitted...
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

14. Delete the remaining "c" files in your home directory. Once more we'll use echo before issuing a destructive command.

```
[student@stationX ~]$ echo *c_[1-3]
...output omitted...
[student@stationX ~]$ rm *c_[1-3]
[student@stationX ~]$ ls
a_reports  december  november  october  Projects september
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

Unit 5

Standard I/O and Pipes

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Know how to redirect I/O channels to files
- Know how to connect commands using pipes

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat Inc
All rights reserved

Standard Input and Output

- Linux provides three I/O channels to processes
  - Standard input - keyboard is default
  - Standard output - terminal window is default
  - Standard error - terminal window is default

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used. copied. or distributed please email <training@redhat .com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Input/Output Streams

One of the most important features of Linux (and UNIX) is the streaming nature of data known as standard input, standard output, and standard error In general, this allows the input from a program to come from any source, and the output to go to any source. In addition, the output from one command can be fed directly to the input of another command.

Standard output, by default, is the screen or terminal window Most commands send their output to standard output without explicitly being told to do so. When standard output is sent to a destination other than the screen, such as a file, one is *redirecting* standard output.

Standard input defaults to the keyboard. Most commands use files as their source of input, but will accept standard input from other sources, via redirection

A third data stream is standard error This is a secondary output stream that carries warnings, usage messages, error messages and other "out-of-band" information, in an output stream distinct from standard output This error stream is normally sent to the screen, but may be redirected elsewhere.

These streams are abbreviated as *stdin* (called file descriptor number 0), *stdout* (file descriptor number 1), and *stderr* (file descriptor number 2) The fact that there are two output channels allows separation of error messages from normal output. For example, error messages could be saved in a file with the normal output going to the monitor.

Redirecting Input and Output

- Standard Input, Output, and Error can be redirected
  - Shell redirection operators allow I/O channels to be redirected to/from a file
  - Pipes allow output channels to be redirected to the input of other programs

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Altering I/O Sources and Destinations

The standard output of commands, which ordinarily displays on the terminal, can be redirected into a file or piped into another command.

Standard error, which also ordinarily displays on the terminal, can be redirected into a file Although it is also possible to pipe standard error into a file using some fairly complex syntax, this is generally not done

Standard input, ordinarily coming from the keyboard, can be redirected from a file. More commonly, the standard output of one command can be piped into the standard input of another command

Common Redirection Operators

command* > *file - Direct standard output of *command* to *file*:

command* >> *file - Append standard output of *command* to *file*

command* < *file - Send *file* as input to *command*

command* 2> *file - Redirect error messages from *command* to *file*

command* 2>> *file - Append error messages from *command* to *file*

Piping

command1* | *command2 - "Pipe" the standard output of *command1* into the standard input of *command2*

Redirecting Output

- To demonstrate I/O redirection we will use the **find** command

  `$ find /etc -name passwd`

- This command will search for all files named `passwd` in `/etc` and its subdirectories.
- By default both the standard output and standard error are displayed on the screen.

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied. or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Example Output

```
[student@stationX ~]$ find /etc -name
passwd
/etc/passwd
find: /etc/default: Permission denied
/etc/pam.d/passwd
```

The specific error messages you receive may vary depending on the software installed on your system

Redirecting Standard Output

- Redirect standard output with >

```
$ find /etc -name passwd > findresult
```

- Standard error is still displayed on the screen

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Example output with stdout redirected to a file and stderr displayed to the terminal:

```
[student@stationX ~]$ find /etc -name passwd > findresult
find: /etc/default: Permission denied
[student@stationX ~]$ cat findresult
/etc/passwd
/etc/pam.d/passwd
```

Copyright © 2006 Red Hat Inc.
All rights reserved

Overwriting vs Appending

- If the target exists, the file will be overwritten
- To append data to an existing file, use >> to redirect instead of >

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Redirect output and create/overwrite the output file

```
[student@stationX ~]$ find /etc -name
passwd > output
find: /etc/default: Permission denied
```

Redirect output and append to the output file

```
[student@stationX ~]$ find /etc -name passwd >> output
find: /etc/default: Permission denied
find: /etc/cups/certs: Permission denied
```

View the output file

```
[student@stationX ~]$ cat output
/etc/passwd
/etc/pam.d/passwd
/etc/passwd
/etc/pam.d/passwd
```

Redirect output and overwrite the output file

```
[student@stationX ~]$ find /etc -name passwd > output
find: /etc/default: Permission denied
```

View the output file

```
[student@stationX ~]$ cat output
/etc/passwd
/etc/pam.d/passwd
```

Copyright © 2006 Red Hat Inc
All rights reserved.

Redirecting Standard Error

- Redirect standard error with 2>
- Example: redirect standard error to a file:

```
$ find /etc -name passwd 2> finderrors
```

- Standard output is displayed on the screen
- Redirect further standard error, appending to the same file, with 2>>

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Redirecting stderr to the file

```
[student@stationX ~]$ find /etc -name passwd 2> finderrors
/etc/passwd
/etc/pam.d/passwd
```

View the file

```
[student@stationX ~]$ cat finderrors
find: /etc/default: Permission denied
find: /etc/cups/certs: Permission denies
```

Redirect some more stderr to the file

```
[bob@station2 tmp]$ find /tmp -name passwd 2>> finderrors
```

View the `finderrors` file

```
[student@stationX ~]$ cat finderrors
find: /etc/default: Permission denied
find: /tmp/orbit-root: Permission denied
```

Copyright © 2006 Red Hat Inc
All rights reserved

Redirecting Both Standard Output and Error

- Redirection of standard output and standard error can be performed simultaneously:

```
$ find / -name passwd 2> errs > results
```

- Each I/O channel can be redirected to different files, or to the same file:

```
$ find / -name passwd > alloutput 2>&1
```

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Redirect stderr to the errs file and redirect stdout to the results file

```
[student@stationX ~]$ find /etc -name passwd 2> errs > results
```

View the errors file

```
[student@stationX ~]$ cat errs
find: /etc/default: Permission denied
```

View the results file

```
[student@stationX ~]$ cat results
/etc/passwd
/etc/pam.d/passwd
```

Redirect both stderr and stdout to the alloutput file

```
[student@stationX ~]$ find /etc -name passwd > alloutput 2>&1
```

View the output file

```
[student@stationX ~]$ cat alloutput
/etc/passwd
find: /etc/default: Permission denied
/etc/pam.d/passwd
```

Copyright © 2006 Red Hat Inc.
All rights reserved.

Redirecting Input

- Redirect standard input with <
- Some commands can accept data redirected to stdin

```
$ tr 'A-Z' 'a-z' < .bash_profile
```

  - This command will translate the uppercase characters in `.bash_profile` to lowercase

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Redirecting the standard input

Many Linux commands, like **cat** and **tr**, will take their input from a file if one is given as an argument. In lieu of a filename, such commands can often also accept the contents of a file redirected directly to the standard input. Note that the following two commands produce the same output:

```
[student@stationX ~]$ cat filename.txt
Hello, World!
[student@stationX ~]$ cat < filename.txt
Hello, World!
```

Copyright © 2006 Red Hat, Inc
All rights reserved

Using Pipes To Connect Processes

- Pipes (the | character) let you redirect output from one command to become the input to another command

```
$ ls /usr/lib | less
```

- Can create *pipelines* - a powerful feature of Linux

```
$ cut -f1 -d: passwd | sort -r | less
```

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Understanding pipes

Two of the basic tenets of UNIX philosophy are: make small programs that do one thing well, and expect the output of every program to become the input to another, as yet unknown, program The use of pipes let you leverage the effects of these two design principles to create extremely powerful chains of commands to achieve your desired result Most command-line programs that operate on files can also accept input on standard in, and output to standard out. Some commands like tr are designed to only operate within a pipe (can not operate on files directly)

Any command that writes to standard output can be used on the left-hand side of a pipe.

Any command that reads from standard input can be used on the right-hand side of a pipe

Multiple commands can be chained together with pipes.

Example output

```
[student@stationX man]$ pwd
/usr/share/man
[student@stationX man]$ ls -C | tr 'a-z' 'A-Z'
MAN1  MAN3  MAN5  MAN7  MAN9  PI_BR   WHATIS
MAN2  MAN4  MAN6  MAN8  MANN  TMAC.H
```

In the example above, lower case letters of the alphabet from the file listing are converted to upper case letters

Useful Pipe Targets

- **less** displays input one page at a time

```
$ ls -l | less
```

- **mail** sends input via email:

```
$ ls -l | mail -s "Files" bob@example.com
```

- **lpr** sends input to the printer

```
$ ls -l | lpr
```

- **xargs** converts input to argument list

```
$ cat files_to_delete.txt | xargs rm -f
```

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

More pipes

Earlier we demonstrated using the **less** command to display a text file one screenfull at a time. But what if the output of a command takes up more than a screen? You can also use **less** in these situations to buffer and view the command s output by piping the command into it

The traditional **mail** command is rather old fashioned for a mailer. But, it is extremely useful for mailing the standard output of a command. The **-s** option allows you to specify a subject for the email

You can also pipe output to the **lpr** command This will send the output to your system's default printer. If you want to use a non-default printer you can specify one with the **-P** option:

All of the above examples demonstrate sending the output of one command to the standard input of another command. But what if you wanted to convert the output of one command into part of the command line for another? For example, in the example shown in this slide we have a file called `files_to_delete.txt` that contains a list of filenames. Suppose it contained the following files:

```
/home/student/letter.txt
/home/student/images/pic.jpg
```

The command shown in the slide would combine the lines of this file into a space-delimited list of arguments to the **rm -f** command In other words, the command would be equivalent to running:

```
[student@stationX ~]$ rm -f /home/student/letter.txt /home/student/images/pic
```

Copyright © 2006 Red Hat Inc
All rights reserved.

tee

- Lets you "T" a pipe: redirect output to a file while still piping it to another program

```
$ set | tee set.out | less
```

  - In example, output from **set** is written to file `set.out` while also being piped to **less**

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Uses of tee

tee is useful to save the output at various stages from a long sequence of pipes. For example:

```
[student@stationX ~]$ ls -lR /etc | tee stage1.out | sort | tee stage2.out ↵
| uniq -c | tee stage3.out | sort -r | tee stage4.out | less
```

Copyright © 2006 Red Hat Inc
All rights reserved.

End of Unit 5

- Questions and Answers
- Summary
  - Standard I/O channels
  - File redirection
    - Standard input (<)
    - Standard Output (>)
    - Standard Error (2>)
  - Pipes

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat Inc
All rights reserved

Lab 5
Standard I/O and Pipes

Goal:	Become familiar with standard input and output and pipes as implemented on a Red Hat Linux system
Estimated Duration:	30 minutes
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*

Copyright © 2006 Red Hat, Inc
All rights reserved

Sequence 1: Standard Input and Output

Instructions:

1. Open your favorite text editor and create two files to use later in the lab:

   `packages1.txt` should contain the following eight lines:

   ```
   amanda
   galeon
   metacity
   mozilla
   postgresql
   procinfo
   rpmfind
   squid
   ```

   `packages2.txt` should contain the following six lines:

   ```
   anaconda
   openssh
   gnome-core
   samba
   sendmail
   xscreensaver
   ```

2. **cat** is the simplest Linux text filter. Its job is to take its input - from a file whose name is supplied as an argument on the command line, or from standard input - and send it, unchanged, to standard output. Test **cat** with `packages1.txt` now.

   ```
   [student@stationX ~]$ cat packages1.txt
   amanda
   galeon
   metacity
   mozilla
   postgresql
   procinfo
   rpmfind
   squid
   ```

Copyright © 2006 Red Hat, Inc.
All rights reserved

3. If **cat** is supplied with no arguments, it expects to receive standard input This means that if you type `cat` at the shell prompt and press *Enter*, nothing appears to happen. In reality, cat is patiently watching standard input, waiting for input to arrive. If you type some characters and press *Enter*, cat sends the typed input to standard output -- effectively echoing back what was typed. To terminate the cat command, issue a *Ctrl-d* from the keyboard. This is the universal end-of-input signal

```
[student@stationX ~]$ cat
Type some sample text, then press return.
Type some sample text, then press return.
Ctrl-d
```

4. Most Linux text-processing commands are implemented as filters - that is, they can read standard input, do something to it, then send it to standard output These commands behave exactly like **cat**, but their output differs in some way from their input.

   tr, introduced in the unit, is such a filter If you supply **tr** with two strings as arguments, it reads from standard input, translating the letters in the first string to letters in the second string, and writes the translated string to standard output.

   Repeat the previous example, using **tr** instead of cat. Supply **tr** with arguments designed to change all vowels in its input to upper case

```
[stdent@stationX ~]$ tr 'aeiou' 'AEIOU'
Type some sample text, then press return.
Ctrl-d
TypE sOmE sAmplE tExt, thEn prEss rEtUrn.
```

Copyright © 2006 Red Hat, Inc
All rights reserved.

5 To specify to the shell that output from a command should not be sent to standard output, but should instead be redirected to a file, redirect standard output with the **>** directive.

Repeat the first **cat** example, redirecting standard output to packages1.catfile. This places everything that would have gone to the screen into `packages1.catfile`, effectively making a copy of `packages1.txt`. **cat** the output file, and verify that it contains the same content as the original file with **diff** and **ls**.

```
[student@stationX ~]$ cat packages1.txt > packages1.catfile
[student@stationX ~]$ cat packages1.catfile
...output omitted...
[student@stationX ~]$ diff packages1.txt packages1.catfile
...output omitted...
[student@stationX ~]$ ls -l packages1*
...output omitted...
```

6. To append an existing file with the contents of another file, use the **>>** directive.

Append `packages1.catfile` with `packages2.txt`, and examine the results

```
[student@stationX ~]$ cat packages2.txt >> packages1.catfile
[student@stationX ~]$ cat packages1.catfile
...output omitted...
```

7. If no filename argument is passed to **cat**, and standard input is redirected to a file, everything typed until input is terminated with *Ctrl-d* will be redirected to the file. This is an easy way to create a new text file

```
[student@stationX ~]$ cat > typedin.txt
This time, when text is typed at the keyboard,
it is redirected to the file typedin.txt.
Ctrl-d
[student@stationX ~]$ cat typedin.txt
This time, when text is typed at the keyboard,
it is redirected to the file typedin.txt.
```

Copyright © 2006 Red Hat Inc
All rights reserved

8. Repeat the previous step, substituting a **tr** command for **cat**.

```
[student@stationX ~]$ tr 'aeiou' 'AEIOU' > trfile.txt
This time, when text is typed at the keyboard,
it is not echoed back to the screen with the translations made.
Instead, it is redirected to the file trfile.txt.
Ctrl-d
[student@stationX ~]$ cat trfile.txt
ThIs tImE, whEn tExt Is typEd At thE kEybOArd,
It Is nOt EchOEd bAck tO thE scrEEn wIth thE trAnslAtIOns mAdE.
InstEAd, It Is rEdIrEctEd tO thE fIlE trfIlE.txt.
```

9. Use **set -o** to display the current setting for the bash noclobber option (off). Verify that you can indeed "clobber" files when redirecting output to a file.

```
[student@stationX ~]$ set -o
...output omitted...
[student@stationX ~]$ ls -l /etc/passwd > trfile.txt
[student@stationX ~]$ cat trfile.txt
-rw-r--r--  1 root root 3911 Sep 28 13:06 /etc/passwd
```

10. Use **set** to modify the noclobber option, then verify its operation:

```
[student@stationX ~]$ set -o noclobber
[student@stationX ~]$ echo "new contents" > trfile.txt
bash: trfile.txt: cannot overwrite existing file
```

11. **cat** will accept either a filename as an argument, or standard input redirected from a file. Test with the following two commands:

```
[student@stationX ~]$ cat packages1.txt
...output omitted...
[student@stationX ~]$ cat < packages1.txt
...output omitted...
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

12. Standard output and input can both be redirected, as in the following example. Direct the input to **tr**, as above, from the `packages1.txt`, but this time redirect standard output. No output is sent to the screen - it is instead stored in the file `packages1.trfile.txt`.

```
[student@stationX ~]$ tr 'aeiou' 'AEIOU' < ↙
packages1.txt > packages1.trfile.txt
[student@stationX ~]$ cat packages1.trfile.txt
AmAndA
gAlEOn
mEtAcIty
mOzIllA
pOstgrEsql
prOcInfO
rpmfInd
sqUId
```

Copyright © 2006 Red Hat Inc.
All rights reserved.

Unit 6

Users, Groups, and Permissions

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Objectives

Upon completion of this unit, you should:

- Understand the Linux security model
- Know the purpose of user and group accounts
- Be able to read and set file permissions

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat Inc.
All rights reserved

The Linux Security Model

- Users and groups are used to control access to files
- Users *log in* by supplying user name and password
- Every file is owned by a user and associated with a group
- Every process has an owner and group affiliation, and can only access the resources its owner or group can access

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The Linux Security Model

Every file on a Linux system is owned by a user, and users cannot change or even read each others' files without being given permission. A user's identity is established at login time when the user gives a login name and password

Because of this, and because unprivileged users do not normally operate with root-level access, Linux is significantly less susceptible to the viruses that plague other operating systems. Users of systems with less rigorous security models operate with system-level access, which allows malicious programs free reign

Copyright © 2006 Red Hat Inc
All rights reserved

Users

- Every user is assigned a unique User ID number (*uid*)
- Users' names and uids are stored in `/etc/passwd`
- Users are assigned a home directory and a program that is run when they log in (usually a shell)
- Users cannot read, write or execute each others' files without permission

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Users

In the earliest years of computing, computers were very large and expensive The concept of user accounts was created to allow many individuals to share these precious computing resources.

Every person that logs into the computer is considered a user. This user has a number of characteristics, the most important being a user name and a user identification number, or uid. Both should be unique on the system User names and uids are stored in the `/etc/passwd` file Other fields in this file include the user's real name and the user's home directory.

Users have full access to their home directories. That is, they can create and remove files and directories as they please, and can organize their files in any way they desire, subject only to limitations such as disk quotas. Typically, users will have limited or no access to other directories on the system, although there are some exceptions (`/tmp`, for example). A user's ability to gain access to files or directories depends on the permissions of the files, as well as the user's identity and the user's group affiliations, to be discussed on the next slide.

Copyright © 2006 Red Hat inc
All rights reserved

Groups

- Users are assigned to groups
- Each group is assigned a unique Group ID number (*gid*)
- gids are stored in `/etc/group`
- Each user is given their own private group
  - Can be added to other groups for additional access
- All users in a group can share files that belong to the group

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Groups

Sometimes users need to collaborate. This can be accomplished by having users assigned to groups and setting appropriate group permissions for files or directories.

Every user is a member of at least one group, and possibly more. As with users, groups have group names and group identification numbers, gids. The group names and gids are stored in the `/etc/group` file.

User Private Group Scheme

By default, a user belongs to a group that is named the same as their username. That is, user `digby` is a member of group `digby` and, by default, is the only member of that group. This system can be abandoned by system administrators when they set up accounts and so this may not be the case at your location.

Primary Group

A user's primary group is defined in the `/etc/passwd` file and secondary groups are defined in the `/etc/group` file. The primary group is important because files created by this user will inherit that group affiliation. The primary group can temporarily be changed by running **newgrp *`groupname`***, where *`groupname`* is one of the user's secondary groups. The user can return to their original group by typing **exit**.

Copyright © 2006 Red Hat Inc.
All rights reserved

The *root* user

- The *root* user: a special administrative account
  - Sometimes called the "superuser"
  - `root` has complete control over the system
    - and an unlimited capacity to damage it!
  - Do not work as `root` unless you need to
    - Normal ("unprivileged") users' potential to do damage is more limited

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The Super User

The `root` user, sometimes called the "superuser", has unlimited access to all the files, devices, and programs on the system. To protect the system from accidental damage, it is important for system administrators to use a normal user account for as much work as possible

When operating from a normal user account, a mistyped command cannot damage the system. When operating as root, a small typo when using **rm**, for example, can irretrievably delete all the files on the system.

Copyright © 2006 Red Hat Inc
All rights reserved

Linux File Security

- File and directories have permissions to determine users' access-levels
- Permissions are set for:
  - the *owner* of the file (called the "user", arguably a misnomer)
  - the *group* members
  - all *others*
- Permissions that are set are called *read*, *write*, and *execute* permissions

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Permission Types

- Four symbols are used when displaying permissions:
  - `r`: permission to read a file or list a directory's contents
  - `w`: permission to write to a file or create and remove files from a directory
  - `x`: permission to execute a program or change into a directory and do a long listing of the directory
  - `-`: no permission (in place of the `r`, `w`, or `x`)

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

File Permissions

Each of the standard permission types can be used to a restrict access for the file's user, access for the file's group, and access for everyone else.

Permissions on files for any particular user category are as follows:

read permission means the contents of the file can be examined with a command such as cat or less.

write permission means the file can be edited and saved

execute permission means the shell will attempt to execute the file when its name is entered as a command

Permissions on directories for any particular user category are as follows:

Directory Permissions

read permission means the contents of the directory can be listed with ls.

write permission means that files may be created in that directory.

execute permission means that the user can cd to that directory and do a long listing (read permission without execute permission permits a listing, but not a long listing).

A file may be removed by anyone who has write permission to the directory in which the file resides regardless of the ownership or permissions on the file itself

Copyright © 2006 Red Hat, Inc
All rights reserved.

Examining Permissions

- File permissions may be viewed using **ls -l**

```
$ ls -l /bin/login
-rwxr-xr-x 1 root root 19080 Apr 1 18:26 /bin/login
```

- File type and file access permissions are symbolized by a 10-character string

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Examining Permissions

The **ls -l** command displays, among other information, the permissions of the file. The first character of the long listing is the file type. The next nine characters are the permissions, explained in succeeding pages

Copyright © 2006 Red Hat Inc
All rights reserved

Interpreting Permissions

```
-rwxr-x--- 1 andersen trusted 2948 Oct 11 14:07 myscript
```

- Read, Write and Execute for the owner, `andersen`
- Read and Execute for members of the `trusted` group
- No access for all others

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Interpreting Permissions

The file shown at the top of this slide can be read, written and executed (`rwx`) by the user `anderson`, read and executed (`r-x`) by members of the group `trusted` and has grants no permissions (`---`) to anyone else. Some more examples are shown below.

Given the following file permissions:

```
-rwxr-xr--   1 fred   fred    26807  Mar  8 22:55  penguin
-rw-r--r--   1 mary   admin    1601  Mar  5 22:36  redhat
-rw-rw-r--   1 root   staff    8671  Mar  8 22:59  tuxedo
```

And the following group memberships:

User	Primary Group	Secondary Groups
fred	fred	staff
mary	mary	staff,admin

The following will be true:

- `penguin` can be read, written, and executed by `fred`, but only read by `mary`;
- `redhat` can be read and written by `mary`, but only read by `fred`;
- `tuxedo` can be read and written by both `mary` and `fred`

Copyright © 2006 Red Hat Inc
All rights reserved

Examining Directories

- The first character in the long listing distinguishes directories (d) from regular files (-)

```
drwxr-xr-x 2 root root  4096 Apr 20 18:13 /bin/
-rwxr-xr-x 1 root root 28596 Feb 18  2003 /bin/df
```

- Other file type indicators exist

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Linux Process Security

- When a process accesses a file the user and group of the process are compared with the user and group of the file
  - If the user matches the user permissions apply
  - If the group matches, but the user doesn't, the group permissions apply
  - If neither match, the other permissions apply

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Process Security

Every process runs as a under the authority of a particular user and with the authority of one or more groups; this is called the process's security context When a process tries to access a file, the security context of the process is matched against the owner and group affiliation of the file

If the user running the process is also the owner of the file, the user permissions apply, regardless of the group and other permissions.

Otherwise, if the process is a member of the file's group, the group permissions apply, regardless of the other permission

If the process is affiliated with neither the owner of the file nor the group of the file, then the other permissions apply.

Note that it is possible to make the owner's permissions more restrictive than the group and other permissions (or to make the group permissions more restrictive than the other permission), but this is odd and rarely done.

Copyright © 2006 Red Hat Inc
All rights reserved

Changing Permissions - Symbolic Method

- To change access modes:
  - **chmod [-R]** *`mode file`*
- Where *mode* is:
  - **u**,**g** or **o** for user, group and other
  - + or - for grant or deny
  - **r**, **w** or **x** for read, write and execute
- Examples:
  - **ugo+r**: Grant read access to all
  - **o-wx**: Deny write and execute to others

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

chmod Syntax

The **chmod** command changes access mode for files and directories The **chmod** command takes a permission instruction followed by a list of files or directories to change. The permission instruction can be issued either symbolically (the symbolic method) or numerically (the numeric method).

Using the symbolic method, the permission expression contains three fields: an indicator of *who* has access to the file, an *operator* for selecting how the permissions should be changed, and a *permission*. If no who value is given, then the permission is added or removed for user, group and other. Multiple, comma separated operations can be given in a single command.

who may be	*operator* may be	*permissions* may be
u User who owns the file	+ Add a permission	**r** Read
g Users in the file's Group	- Remove a permission	**w** Write
o Other users	= Assign a permission	**x** Execute or cd
a All three categories		**s** Set user ID bit or group
		t Sticky bit (for directories)

Examples:

chmod u+w,go-w .bashrc grants write access to owner but denies it to group and other

chmod u=rw .bashrc sets user permissions to read and write, with execute turned off, regardless of the current permissions

chmod +r .bashrc makes the file world-readable

A useful option to **chmod** is **-R** (recursive). This option tells **chmod** to traverse an entire directory tree to change the permissions of all its files and subdirectories The **s** and **t** permissions will be discussed in a later unit.

Copyright © 2006 Red Hat. Inc.
All rights reserved

Changing Permissions - Numeric Method

- Uses a three-digit mode number
  - first digit specifies owner's permissions
  - second digit specifies group permissions
  - third digit represents others' permissions
- Permissions are calculated by adding:
  - 4 (for read)
  - 2 (for write)
  - 1 (for execute)
- Example:
  - **chmod 640 myfile**

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Numerical Mode Setting

To change all the permissions on a file at once, it is often easier and quicker to use the numeric method.

In this method, the first argument to the **chmod** command is a three digit number representing the permissions that are set. In this method, the read permission has a value of four, the write permission has a value of two, and the execute permission has a value of one. Add together the permissions you wish to set for user, group, and others. Examples:

chmod 664 ***file*** grants read/write to the owner and group, read-only to others

chmod 660 ***file*** grants read/write to the owner and group; no permissions for others

chmod 600 ***file***: grants read/write to the owner; no permissions set for the group and others

chmod 444 ***file***: grants read-only to all

With directory permissions, when you set the read permission, you almost always want to set execute permission. Examples:

chmod 755 ***dir***: grants full permissions to the owner, read and execute to group and other.

chmod 770 ***dir***: grants full permissions to the owner and group, no permissions to others

chmod 700 ***dir***: grants full permissions to the owner, no permissions to the group or others

chmod 555 *dir*: grants read and execute permission set to all.

421
RWX

Copyright © 2006 Red Hat, Inc.
All rights reserved

Changing Permissions - Nautilus

- Nautilus can be used to set the permissions and group membership of files and directories.
  - In a Nautilus window, right-click on a file
  - Select Properties from the context menu
  - Select the **Permissions** tab
- **Text view** and **Number view** of the permissions are updated as you make changes.

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat Inc
All rights reserved

End of Unit 6

- Questions and Answers
- Summary
  - All files are owned by one user and one group
  - The mode of a file is made up of three permissions: those of the user, the group and all others
  - Three permissions may be granted or denied: read, write and execute

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Lab 6
File Permissions

Goal: Become familiar with the functions, syntax, and use of several essential file permission modification commands, and how you can combine these commands together in useful ways to accomplish common user tasks.

Estimated Duration: 45 minutes

System Setup: A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.

Copyright © 2006 Red Hat Inc
All rights reserved

Sequence 1: Practice defining file permissions

Instructions:

1 What is the symbolic representation (eg rwxr-xr-x) of each of the following numeric permissions?

644 rw- r-- r--
755 rwx r-x r-x
000 --- --- ---
711 rwx --x --x
700 rwx --- ---
777 rwx rwx rwx
555 r-x r-x r-x
111 --x --x --x
600 rw- --- ---
731 rwx -wx --x

2. Given a file with permissions 755, what commands would change the permissions to r-xr--r--?

chmod 755 file / chmod u=rwx, g=rx, o=rx

3. You've just downloaded a file that you know is trustworthy from the Internet, and you want to execute it. What step must you perform before you can run it? List two different ways to perform that step.

chmod a=x file
chmod u=x file
chmod u+x file

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 1 Solutions

1. **644** rw-r--r--
   755 rwxr-xr-x
   000 ---------
   711 rwx--x--x
   700 rwx------
   777 rwxrwxrwx
   555 r-xr-xr-x
   111 --x--x--x
   600 rw-------
   731 rwx-wx--x

2. There are several possible solutions:

   chmod 544 filename

   chmod u-w,go-x filename

   chmod u=rx,go=r filename

3. The command must be made executable Either one of the following commands will accomplish this (and other commands will also work):

   chmod +x cmdname

   chmod 755 cmdname

Copyright © 2006 Red Hat Inc
All rights reserved

Unit 7

vi and vim Editor Basics

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Objectives

Upon completion of this unit, you should:

- Know the three primary modes of **vi** and **vim**
- Be able to navigate text and enter Insert mode
- Be able to change, delete, yank, and put text
- Be able to undo changes
- Be able to search a document
- Be able to save and exit

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat Inc
All rights reserved

Overview of vi and vim

- **vi**: the "visual editor", standard Linux and Unix editor
- **vim**: the "vi improved" editor, standard Red Hat editor
- On Red Hat operating systems, the **vi** command invokes **vim**
- Derived from earlier Unix editors
  - **ed -> ex -> vi -> vim**

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The history of vi

The **vi** editor is the standard editor under Linux and Unix systems, with a heritage almost as long as Unix itself Red Hat ships both the traditional **vi** editor and the newer **vim** editor, the "vi improved' editor.

When Unix was originally developed, the common user interface was the teletype, a typewriter-like device that printed output on side-holed paper The editor in use at the time was **ed**, a line editor. When video terminals first became widely used, first the **ex** line editor, providing extensions to **ed** taking advantage of the video nature of the new devices, was created, followed by **vi** screen editor. The **ex** commands remain in **vi**, particularly the search-and-replace, saving, and exiting functions

The **vim** editor is a more recent version of **vi**, providing extensive improvements, primarily for power users On Red Hat operating systems, the **vi** command is aliased to **vim**.

Copyright © 2006 Red Hat Inc
All rights reserved

Starting vi and vim

- To start **vim**:
  - **vi** ***filename***
  - If the file exists, the file is opened and the contents are displayed
  - If the file doesn't exist, **vi** creates it when the edits are saved for the first time
- To use **vi** instead:
  - **unalias vi** or,
  - **\vi**

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Running vi

To edit an existing file, run the command:

vi filename

Edits performed on this file will be made in a buffer until the file is saved; that is, you can undo individual changes or even abandon all changes until you save the changes. To create a new file with **vi**, use the same command, giving a new filename as an argument to **vi**

When invoking **vi** in this way, you actually invoke **vim**, as **vi** is an alias to **vim**. If you wish to call **vi** itself, you can call it on the fly as:

```
[student@stationX ~]$ \vi
```

Alternatively, if you wish to use **vi** as your default editor, place the following command in one of your startup scripts (more on startup scripts in a later unit):

```
[student@stationX ~]$ unalias vi
```

Some useful options that you can use with **vi** include:

-m *filename*	makes the opened file non-modifiable
-R *filename*	makes the opened file non-modifiable, but can be overridden with **:w!**
-n *filename*	does not use a swap file for backup (useful for floppies)
-r *filename*	recovers data from a swap file after a crash
-x *filename*	encrypts the file when saving and decrypts when editing

Copyright © 2006 Red Hat Inc.
All rights reserved

Three Modes of vi and vim

- Command mode
  - Cursor movement
  - Change, delete, yank, put, search
- Insert mode
  - Type in new text
  - Return to command mode with *Escape*
- ex mode
  - Configuring, exiting, saving
  - Search and replace

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

An introduction to modal editors

The **vi** editor operates in three "modes", three different states of operation It is vital that you be aware of the current mode, as different sorts of things happen in each mode. The three modes are called *command mode*, *insert mode* and *ex mode*.

When you first enter **vi**, you will be in command mode. This is considered the "home" mode, the mode to which you will return when you are otherwise not performing some specific action. When in command mode, the keystrokes you enter do not become part of the text, but rather are interpreted as command mode commands. Among the actions you can take from command mode are:

- Moving the cursor, entering insert mode
- Changing text, deleting text
- Copying text (called *yanking*)
- Pasting copied or deleted text (called *put*)
- Entering ex mode

In insert mode, your keystrokes are actually data entered into your document, rather than commands. In coming slides, you will learn how to enter insert mode and how to return to command mode from insert mode.

Copyright © 2006 Red Hat, Inc
All rights reserved.

The final mode is ex mode. In this mode, you can enter extended commands. Among the extended commands are saving, exiting, and search-and-replace commands. To enter ex mode, type **:** while in Command Mode.

Cursor Movement
By Character

- **h** left
- **j** down
- **k** up
- **l** right
- arrow keys also work

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Moving by character

In command mode, one of the most important actions that you will want to take is to move the cursor. Not only are cursor movements important in themselves, but they are also the basis for much of the rest of the**vi** and **vim** command vocabulary

The commands above will move the cursor about the file. There are many more cursor movements, but these are the most important and useful ones.

The **h**, **j**, **k**, and **l** movements move the cursor left, down, up, and right, respectively. These keys can be typed with three fingers of the right hand The arrow keys also work, but note that if you become adept at using the **h**, **j**, **k**, and **l** keys, you can move the cursor without moving you hands from the base row of the keyboard.

Moving by word

The **w** and **b** keys move forward and backward a word at at time. A word is defined as a series of letters of the alphabet and numbers uninterrupted by white space or punctuation If the cursor is on a punctuation character, the word is terminated by white space or a letter of the alphabet or number. For example:

"Hello!", she said.

Words on this line are:

"	Hello	!",	she	said	.
1	2	3	4	5	6

Six words. Note word number three: a series of punctuation marks.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Cursor Movement
In larger chunks

	word	sentence	paragraph
Move left one....	w	)	}
Move right one....	b	(	}

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Moving by word

The **w** and **b** keys move forward and backward a word at at time A word is defined as a series of letters of the alphabet and numbers uninterrupted by white space or punctuation If the cursor is on a punctuation character, the word is terminated by white space or a letter of the alphabet or number For example:

"Hello!", she said.

Words on this line are:

"	Hello	!",	she	said	.
1	2	3	4	5	6

Six words Note word number three: a series of punctuation marks.

Moving by paragraph and sentence

You can also move by larger chunks such as paragraphs and sentences) and (move forward and back one sentence, a sentence being defined as a text followed by a blank line or punctuation. } and { move forward and back one paragraph, a paragraph being defined as text followed by a blank line.

Copyright © 2006 Red Hat Inc.
All rights reserved

Entering Insert Mode

- `a` appends after the cursor
- `i` inserts before the cursor
- `o` opens a line below
- `A` appends to end of line
- `I` inserts at beginning of line
- `O` opens a line above

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Editing text in Insert mode

Many commands will take you into insert mode. The slide above lists six common ways to do so.

The **a** command will place you in insert mode, allowing you to *append after* the cursor The letter **a** will not appear in your document, but all other characters that you type will appear, until you exit insert mode To exit insert mode, hit the *Esc* key

The **i** command is similar to the **a** command, but you will *insert* your data *before* the cursor Again, exit insert mode by hitting the *Esc* key.

The **o** command *opens* a line *below* the current line, placing you in insert mode.

Note the relationship between the lower case **a**, **i**, and **o**, and the upper case **A**, **I**, and **O** Whereas the **a** appends after the cursor, the **A** appends at the end of the line The **i** inserts before the cursor; the **I** inserts at the beginning of the line The **o** opens a line below the current line; the **O** opens a line above the current line. Patterns such as these permeate the **vi** and **vim** commands

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Leaving Insert Mode: Esc

- *Esc* takes you from insert mode back to command mode
- Hint: when in trouble, press *Esc* and then press *Esc* again

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Returning to Command mode

As stated earlier, to get from insert mode back to command mode, press the *Esc* key This is worth repeating, not just because of the usefulness of the action, but also because command mode is the "home"; mode, the place to be when you are not otherwise taking some action. Sometimes, you may "get lost": press an incorrect key and not be sure in what mode you reside When this happens, press the *Esc* key and then press it again. By doing this, you can guarantee that, regardless of the mode you were in, you will now be in command mode.

Copyright © 2006 Red Hat, Inc
All rights reserved

Change, Delete, and Yank

	Change	Delete (cut)	Yank (copy)
Line	cc	dd	yy
Letter	cl	dl	yl
Word	cw	dw	yw
Sentence ahead	c)	d)	y)
Sentence behind	c(	d(	y(
Paragraph above	c{	d{	y{
Paragraph below	c}	d}	y}

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Change, Delete and Yank (Replace, Cut and Paste)

At first glance, the chart above may seem intimidating, but note the pattern The characters **w**, **{**, **}**, **(**, and **)** mean the same thing here as in cursor movement The commands **c**, **d**, and **y**, are combined with these to perform an action

The **cc** command will change a line: it will delete the current line and place you into insert mode to enter the replacement, which could be another line, several lines, or just a few characters As before, press *Esc* to return to command mode.

The **cl** command will delete the current character and place you in insert mode; the **cw** command will delete the current word and place you into insert mode The paragraph and sentence commands will operate similarly.

dd deletes the current line, leaving you in command mode The remaining deletion commands operate in a similar fashion

The **yy** command may take some explanation These days, the common name for this action is "copy"; that is, place some text in a buffer without modifying the original data. However, in the original Unix **vi**, and still today in the modern **vim**, the action taken is not called "copy", but rather "yank". A line is yanked into a buffer, presumably to be put (or pasted) back into the document at another location. As before, **yl** yanks a letter, **yw** yanks a word, and the remaining commands yank sentences and paragraphs.

In the next slide, we will learn to put (or paste) yanked data back into the document.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Put (paste)

- Use `p` or `P` to put (paste) copied or deleted data
- For line oriented data:
  - `p` puts the data below the current line
  - `P` puts the data above the current line
- For character oriented data:
  - `p` puts the data after the cursor
  - `P` puts the data before the cursor

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Put (paste)

Just as in **vi** and **vim**, we "yank" instead of "copy", we "put" instead of "paste": take data from a buffer and place it in the document.

The commands to put data from a buffer into the document are `p` and `P`. How these work depend on the nature of the data in the buffer. If the data is line oriented (you yanked a line or a paragraph), the `p` command will open a new line below the current line and place the data there; the `P` command will open a new line above the current line and place the data there. If the data is character oriented (you yanked a letter, word, or sentence), the `p` command will put the data after the cursor and the `P` command will put the data before the cursor

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Undoing Changes

- **u** undo most recent change
- **U** undo all changes to the current line since the cursor landed on the line
- ***Ctrl*-r** redo last "undone" change

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Help! I made mistake!

You will not work with **vi** or **vim** long before needing to undo a change that you made, either incorrectly or unintentionally. To undo the most recent change, use the **u** command. Several **u** commands in a row will undo several previous changes. The **u** will not undo a previous **u**; that is, it will not toggle a change, but rather undo several previous changes.

To undo all successive changes to the current line, use the **U** command.

To redo a change undone by a **u** command, use the ***Ctrl*-r** command.

Searching for Text

- `/text` search downwards for "text"
- `?text` search upwards for "text"
- `n` continue search in the same direction
- `N` continue search in the opposite direction

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Searching a document

To search for a string of text, use the **/** or **?** commands as in the *less* command:

`/dog` search downwards in the file for the string "dog"

`?mouse` search upwards in the file for the string "mouse"

Once the first match is found, you can find the *next* match with the **n** command. To reverse the direction of the search, use the **N** command.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Command-Mode Tricks

- `dtc` deletes from cursor to the letter *c*(does not span lines)
- `5dd` deletes five lines (a number can precede any of the two character change, delete, yank, or put commands)
- `x` deletes a character
- `rc` replaces a character with *c*
- `R` replaces character-for-character until *Esc*

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Some advanced command-mode functions

The **vi** and **vim** editors are tremendously featureful. The list of additional options are legion. The slide above contains just a few helpful tricks, described here:

Typing `dtx` will "delete" from the cursor "to" the character *x*. This command does not span lines. If the final character does not exist on the line, nothing will be deleted

Typing `5dd` will "delete" *five* lines Any number can precede any of the change, delete, yank, or put commands Thus, `3y}` will yank three paragraphs below the current line and `2p` will put the three paragraphs back into the document twice, for a total of six extra paragraphs.

Typing `x` will "delete" a character. The command `dl` also will delete a character, but the `x` command presents a terse method.

Typing `rx` will "replace" the character under the cursor with the character `x` This task can be accomplished in a number of ways. For example `clx`***Esc***, but note that the `rx` command takes only two keystrokes, half the number of the longer method.

Typing `R` will "replace" one character for every character typed until the Esc character is entered. This is an excellent method for doing replacements character for character. Note that using something like `15cl` requires that you actually count the number of characters that you wish to change With the `R` command, you do not need to count the characters in advance.

Copyright © 2006 Red Hat, Inc
All rights reserved

5. The last sentence of the document has been duplicated. Move to the beginning of the sentence with } and) and delete it with **d)**:

```
})))))d)
```

6. Finally, write (save) your corrected file to the disk and exit **vi**:

```
:wq
```

If this does not have the expected effect, press *Esc* twice to enter command mode and try again. If unwanted changes were made, use the **u** key to undo them from command mode before saving.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Unit 8

The Linux Filesystem In-Depth

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Know how to create partitions and file systems
- Understand the function of dentries and inodes
- Know how **cp**, **mv**, and **rm** work at the inode level
- Understand how symbolic links and hard links work
- Know how to access removable media
- Be able to create archives using **tar** and **gzip**

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Partitions and Filesystems

- Disk drives are divided into *partitions*
- Partitions are formatted with *filesystems*, allowing users to store data
  - Default filesystem: ext3, the Third Extended Linux Filesystem HAS JOURNAL
  - Other common filesystems:
    - ext2 and msdos (typically used for floppies)
    - iso9660 (typically used for CDs) + FLASH

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

An overview of filesystems in Red Hat Enterprise Linux

On Red Hat Enterprise Linux systems, disk drives and other storage media typically are divided into partitions. Also typically, a partition is formatted with a filesystem A filesystem is a data structure written to the media that allows users to store and access files.

The default filesystem for Red Hat Enterprise Linux is the Third Extended Linux Filesystem or ext3. It is an enhanced version of ext2 that uses journaling to improve filesystem data integrity Ext2 has been in use on Linux since 1993 making it and its successor, ext3, very stable and reliable Ext2/3 support a variety of advanced features which are not always present in other Unix filesystems such as extended attributes (EAs) and POSIX Access Control Lists (ACLs).

Red Hat Enterprise Linux supports over 20 different filesystem types. Other common filesystems are msdos, which is used for compatibility on floppy disks and iso9660 which is used on CDs. It is also common to format floppies with ext2 because it is more powerful and flexible than msdos and uses less disk space for its data structures than ext3

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Inodes FAT

- An *inode table* contains a list of all files in an ext2 or ext3 filesystem
- An *inode* (index node) is an entry in the table, containing information about a file (the *metadata*), including:
  - file type, permissions, link count, UID, GID
  - the file's size and various time stamps
  - pointers to the file's data blocks on disk
  - other data about the file

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

What is in inode? IS-RILES FILE

ext2 and ext3 file systems keep a list of files in the file system in a table called an inode table An individual entry in the inode table is called an inode. The inode is referenced by its number, the inode number, which is unique within a file system

The inode contains the metadata about files (remember that everything in Linux is a file; the term "file" is being used in this broad manner in the inode discussion) Among the data stored in the inode is:

- the file type
- file permissions
- link count: the number of file names associated with the inode number (more on this later)
- the user ID number of the file owner
- the group ID number of the associated group
- time stamps, including last access time, last modification time, and inode change time
- location of the data on the hard disk
- other metadata about the file

Copyright © 2006 Red Hat, Inc
All rights reserved.

Directories

- The computer's reference for a file is the *inode number*
- The human way to reference a file is by *file name*
- A *directory* is a mapping between the human name for the file and the computer's inode number

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

What is a directory?

We commonly think of a directory as a container for files and other directories. In fact, a directory is a mapping between the file names that humans use to reference files and inode numbers used by the computer to reference files.

When a filename is referenced by a command or application, Linux references the directory in which the file resides, determines the inode number associated with the file name, looks up the inode information in the inode table, and, if the user has permission, returns the contents of the file.

The **ls -i** command displays the inode number:

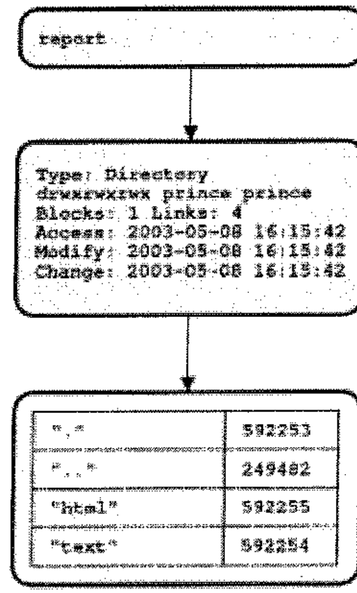
```
[student@stationX ~]$ ls -il
total 28
80788 -rw-r-----   1 student  student     5120 Sep 18 11:26 myData
37777 drwxr-x---   2 student  student     4096 Sep 18 11:25 newStuff
80787 -rw-r-----   1 student  student     1536 Sep 18 11:26 notes
80789 -rw-r-----   1 student  student    11776 Sep 18 11:26 otherStuff
```

The inode number for the `myData` file is 80788.

Inodes and Directories

Name

Associated with an inode by parent directory



Inode Metadata

Properties and a pointer to blocks on disk

Contents

For directories: name/inode list (shown)
For files: arbitrary data

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

cp and inodes

- The **cp** command:
  1. Allocates a free inode number, placing a new entry in the inode table
  2. Creates a dentry, associating a name with the inode number
  3. Copies data into the new file

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

cp in-depth

Consider the nature of the **cp**, **mv**, and **rm** commands in the context of the inode table

When a file is copied to a new name in the same directory, the directory and the inode table get a new entry:

```
[student@stationX ~]$ ls -li penguin
246674 -rw-rw-r--    1 digby   digby          26 Sep 25 20:56 penguin

[student@stationX ~]$ cp penguin tux

[student@stationX ~]$ ls -li penguin tux
246674 -rw-rw-r--    1 digby   digby          26 Sep 25 20:56 penguin
246575 -rw-rw-r--    1 digby   digby          26 Sep 25 20:56 tux
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

mv and inodes

- If the destination of the **mv** command is on the same file system as the source, the **mv** command:
  1. Creates a new directory entry with the new file name
  2. Deletes the old directory entry with the old file name
- Has no impact on the inode table (except for a time stamp) or the location of data on the disk: no data is moved!

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

mv in-depth

The name of the **mv** command works neatly with the metaphor that a directory is a container for files. But, as we now know that a directory is a mapping between file names and inode numbers, the **mv** command must be understood in these terms.

When a file is moved, the underlying file, either as inode entry or as data on the hard disk, does not move. What moves is the entry in a directory. When a file is moved within the same file system, only the directory itself (or directories if moving between directories) are affected:

```
[student@stationX ~]$ ls -li tux
246575 -rw-rw-r--    1 digby   digby       26 Sep 25 20:56 tux

[student@stationX ~]$ mv tux fedora

[student@stationX ~]$ ls -li fedora
246575 -rw-rw-r--    1 digby   digby       26 Sep 25 20:56 fedora
```

The inode number remains the same. The data on the file system is not moved. The inode is not change, except that the inode change time is updated.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

rm and inodes

- The **rm** command:
  1. Decrements the link count, thus freeing the inode number to be reused
  2. Places data blocks on the free list
  3. Removes the directory entry
- Data is not actually removed, but will be overwritten when the data blocks are used by another file

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

rm in-depth

The **rm** command has a broad effect. The entry in the directory is effectively deleted; the inode number is made available; the block locations that the file was using are placed on the free list

The underlying data is not actually removed, however The data will be overwritten eventually when the blocks are reused by some other file, but the data is otherwise unmodified.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Symbolic (or Soft) Links

- A symbolic link points to another file
  - **ls -l** displays the link name and the referenced file

    ```
    lrwxrwxrwx 1 joe joe 11 Sep 25 18:02 pf -> /etc/passwd
    ```
  - File type: **l** for symbolic link
  - The content of a symbolic link is the name of the file that it references
- Syntax:
  - **ln -s filename *linkname***

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Symbolic Links

A symbolic link (or symlink) is a file that points to another file That is, when you read from the symbolic link using, perhaps the **cat** or **less** command, you will actually receive the contents of the underlying file Most actions, in fact, will be performed on the underlying file, with the exception of the **rm** command. Removing a symlink removes the actual link itself, not the underlying file.

To create a symbolic link, use the **ln** command with the **-s** option:

```
[student@stationX ~]$  ln -s /etc/passwd password
```

In this example the file passwd in the current directory points to `/etc/passwd` The **ls -l** command lists the symlink and the file that is being referenced by the link:

```
[student@stationX ~]$ ls -li password  /etc/passwd
30338 -rw-r--r--  1 root    root       1729 Aug 24 11:43 /etc/passwd
33276 lrwxrwxrwx  1 digby   digby        11 Sep 26 09:33 passwd -> /etc/passwd
```

Note a few interesting things about this long listing First, it has its own inode number: a symbolic link is a separate file from the original. Second, the first character of a long listing for a symlink is the letter l: a symlink is a special type of file and so it has its own file type indicator. Third, note the odd permissions of the symbolic link. The permissions on a symlink are irrelevant; the permissions set on the file pointed to by the symlink control access rights. Finally, note the size of the symbolic link. The size in this case is 11: there are 11 characters in `/etc/passwd`. The contents of a symlink is the path name that the symlink references. Therefore, the size of the symlink is always the number of characters in the path name.

Hard Links

- One physical file on the filesystem
- Each link references the file's inode
- File is present in the filesystem as long as at least one link remains
- Cannot span drives or partitions
- Syntax:
  - **ln** *`filename`* [*`linkname`*]

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Hard links

A hard link is one of the more difficult types of files to understand. A hard link is a path name that references an inode: that is, all files are hard linked at least once. The interesting twist in the Linux world is that a file can be hard linked more than once Remember that an individual file is referenced by its inode number, the file name merely being a human convenience for referencing the inode number Because the name of a file is separate from an inode (it is stored in a directory, not in the inode), it is possible to have multiple file names pointing to the same inode number

To create an additional hard link to an existing file, use the **ln** command:

```
[student@stationX ~]$  ln fedora redhat

[student@stationX ~]$  ls -li fedora redhat
246575 -rw-rw-r--    2 digby   digby         26 Sep 25 20:56 fedora
246575 -rw-rw-r--    2 digby   digby         26 Sep 25 20:56 redhat
```

The most notable effect here is that the two files, fedora and redhat, have the exact same inode: there is only one underlying file, but there are two entry points Also note that they are both regular files: an additional hard link is not a separate file type Finally, note the link count. The link count is the number after the permissions and before the user name The link count has been incremented to two because two path names point to the same file. When one path name is deleted, perhaps using the **rm** command, the link count will decrement to one; when the final path name is deleted, the link count is decremented to zero and the file is removed.

Two restrictions to additional hard links should be noted: first, the two file names must be on the same filesystem: because they share an inode number and an inode table is unique to a file system, both must be on the same file system. Second, it is not possible to use the **ln** command to create additional hard links to directories.

Copyright © 2006 Red Hat, Inc
All rights reserved

The Seven Fundamental Filetypes

ls -l symbol	File Type
`-`	regular file
`d`	directory
`l`	symbolic link
`b`	block special file
`c`	character special file
`p`	named pipe
`s`	socket

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Identifying basic file types

In the preceding pages, we have learned about three file types: regular files, directories, and symbolic links In Red Hat Enterprise Linux, there are a total of seven file types, called the Seven Fundamental Filetypes of Linux and Unix. Below is a list of the remaining file types and a brief explanation of each, preceded by the symbol indicating the file type in a long listing:

c *character special file*: in the overview that we stated that everything in Linux is a file, even hardware. Files referencing hardware are not regular files; they are one of two types of special files. Character special files are used to communicate with hardware one character at a time.

b *block special file*: used to communicate with hardware a block of data at a time: 512 bytes, 1024 bytes, 2048 bytes: whatever is appropriate for that type of hardware Run the following command to see a list of block and character special files:

```
[student@stationX ~]$ ls -l /dev | less
....output omitted....
```

p *named pipe*: a file that passes data between processes. It stores no data itself, but passes data between one process writing data into the named pipe and another process reading data from the named pipe. A named pipe can be created using the **mknod** command:

```
[student@stationX ~]$ mknod mypipe p
```

s *socket*: a stylized mechanism for interprocess communications. It is extremely rare for a user or even a system administrator to explicitly create a socket.

NETWORK COMMUNICATION

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Checking Free Space

- **df** - Reports disk space usage df -i - INODE USAGE
  - Reports total kilobytes, kilobytes used, kilobytes free *per file system*
  - **-h** displays sizes in easier to read units
- **du** - Reports disk space usage
  - Reports kilobytes used *per directory*
  - Includes subtotals for each subdirectory
    - **-s** option only reports single directory summary
  - Also takes **-h** option

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*Getting usage summaries with **df***

The **df** and **du** commands are used to determine how much disk space is used. The **df** command reports on a per filesystem basis. It reports total disk space, disk space used, and disk space free The disk space free column is often the most useful information:

```
[student@stationX ~]$ df /opt
Filesystem       1K-blocks      Used Available Use% Mounted on

/dev/hda2         16512384   5134496  10539108  33% /opt
```

With typically large file systems, the units above can be hard to read. The **-h** option used multipliers such as G and M for gigabytes and megabytes, respectively:

```
[student@stationX ~]$  df -h /opt
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda2              16G  4.9G   11G  33% /opt
```

*Getting detailed usage information with **du***

The **du** command reports the number of kilobytes contained by the items within a directory By default, **du** will report on the given directory, plus all subdirectories. The **-s** option can be used to request only the summary directory information. The **-h** option is also available Example:

```
[student@stationX ~]$  du -s /etc
62552   /etc
```

Copyright © 2006 Red Hat Inc
All rights reserved

EJECT (-T) - OPEN CD
-T- CLOSE CD

Removable Media

- *Mounting* means making a foreign filesystem look like part of the main tree.
- Before accessing, media must be mounted
- Before removing, media must be unmounted
- By default, non-root users may only mount certain devices (cd, dvd, floppy, usb, etc)
- Mountpoints are usually under `/media`

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Handling removable media

Before you can access the data on newly inserted removable media (floppy disk, CD, zip disk), the filesystem(s) on the media must first be mounted. Although mounting filesystems is generally a system administrator-only function, non-privileged users can mount removable media in a number of ways

Non-privileged users logged into the console are permitted to mount and unmount removable media using commands such as:

```
[student@stationX ~]$ mount /media/floppy        mount /media/cdrom
umount /media/floppy
umount /media/cdrom
```

Note that if you have a cd writer it will be mounted under `/media/cdrecorder` instead of `/media/cdrom`.

An alternative to mounting disks is to use the **mtools** commands The mtools commands mimic standard DOS commands. The a: drive is generally mapped to your floppy You can list, copy, format, and otherwise manipulate removable media, particularly floppies, using DOS commands preceded by an "m":

```
[student@stationX ~]$ mdir a:
 ...output omitted...
[student@stationX ~]$ mcopy myfile a:
```

These command only work on DOS formatted floppies For a complete list of the mtools commands, run **mtools**.

Copyright © 2006 Red Hat. Inc.
All rights reserved

Mounting CDs and DVDs

- Automatically mounted in Gnome/KDE
- Otherwise, must be manually mounted
  - CD/DVD Reader
    - **mount /media/cdrom**
  - CD/DVD Writer
    - **mount /media/cdrecorder**
- **eject** command unmounts and ejects the disk

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Accessing CDs and DVDs

When using the X Window System, inserting a CD into the drive automatically mounts the CD and adds an icon to the desktop.

The mountpoint associated with a device, which you will need to know when mounting it manually, depends on whether you are using a CD/DVD reader or or writer. Readers are mounted under `/media/cdrom` and writers under `/media/cdrecorder`.

When you are done using the disk you can eject it by either right-clicking on the desktop icon and selecting "Eject" or running the **eject** command from a prompt. If you have a single device, running eject by itself is sufficient If you have multiple devices you will have to include the appropriate device node as an argument

Copyright © 2006 Red Hat Inc.
All rights reserved.

Mounting USB Media

- Detected by the kernel as SCSI devices
  - `/dev/sdaX` or `/dev/sdbX` or similar
- Automatically mounted in Gnome/KDE
  - Icon created in Computer window
  - Mounted under `/media/`*`Device ID`*
    - Device ID is built into device by vendor

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Accessing USB storage

Most vendors give USB disks a label. **fstab-sync** reads that label and automatically puts an entry in `/etc/fstab` for it, mounting it in `/media/`*`label`*.

USB disks are treated as SCSI devices, thus they are referenced as `/dev/sda`, `/dev/sdb`, etc. Normally, USB memory sticks use the first partition (e.g., `/dev/sda1`), but they may use other partition--`/dev/sda4` is common. Check your logs for information regarding USB devices and partitions after you plug in the USB disk.

Like other disks, you can use **fdisk** and **mke2fs** to create partitions and filesystems on USB disks.

Copyright © 2006 Red Hat Inc
All rights reserved

Mounting Floppy Disks

- Must be manually mounted and unmounted
  - **mount** `/media/floppy`
  - **umount** `/media/floppy`
- DOS floppies can be accessed with **mtools**
  - Mounts and unmounts device transparently
  - Uses DOS naming conventions
    - **mdir a:**
    - **mcopy** `/home/file.txt` **a:**

17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*Accessing floppies with **mtools***

An alternative to mounting disks is to use the **mtools** commands The **mtools** commands mimic standard DOS commands While **mtools** can be configured to recognize any device, the only one they recognize automatically is the floppy drive which is mapped to "a:"". You can list, copy, format, and otherwise manipulate the media, using DOS commands preceded by an "m" (**mcopy, mdir**, **mdel**, etc). For a complete list of the **mtools** commands, run **mtools** and/or consult the **mtools** info page. The **mtools** commands only work on DOS-formatted floppies

Copyright © 2006 Red Hat, Inc
All rights reserved

Formatting Floppy Disks

- Two types of format needed to prepare a floppy:
  - A low level format (rarely needed)
    - **fdformat /dev/fd0H1440**
  - A filesystem format, one of:
    - **mkfs -t ext2 /dev/fd0**
    - **mke2fs /dev/fd0**
    - **mkfs -t vfat /dev/fd0**
    - **mformat a:**

18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Low-level formatting

Before a floppy can be used, it needs a low level format and a file system

Virtually all floppies these days already have a low level format. If, for some reason, you receive a floppy that does not contain a low level format, use the **fdformat** command to perform this function:

```
[student@stationX ~]$ fdformat /dev/fd0H1440
```

This instructs the computer to put a low level 1.44MB format on the floppy

Applying filesystems to floppies

More commonly, you may need to put a file system on the floppy The **mkfs** command can do this, but you will want to specify a file system type The **mkfs** command, by default, puts an ext3 file system on the floppy, but because of the overhead of ext3, it is better to use a different file system type. The slide above shows two different ways to place and ext2 file system on a floppy and two ways of placing a DOS compatible file system on the floppy.

Only the superuser or a non-privileged user logged into the system console can run these commands Others do not have permission to do this

Copyright © 2006 Red Hat Inc
All rights reserved.

Why Archive Files?

- Archiving places many files into one target file
  - Easier to back up, store, and transfer
- **tar** - standard Linux archiving command

19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The value of file archives

Archiving files is generally a good idea if you want to back up a few directories or transfer many files over a network. Originally, **tar** was used to create archives on tape devices, hence its name - which stands for *tape archive*.

While **tar** is seldom used today to back up entire filesystems, it is often used to bundle together related files before moving or compressing them

tar archives filenames are usually created with `.tar` filename extensions, although this is not required.

Does Not Compress files

Creating an Archive

- Syntax
  - **tar cvf** ***archive_name files...***
  - **c**: creates a new archive
  - **v**: produces verbose messages
  - **f** *archive_name*: is name of new file
- Options do not need a leading dash

20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*Creating archives with **tar***

While most commands in Linux require a dash before options, a few, such as **tar**, do not

The files to be archived need not be in the same directory. If a directory is specified in the file list, it is archived recursively, that is, its contents are also archived including all subdirectories

The **v** option is not necessary, but shows which files are being added to the archive. Using the **v** option can significantly increase the time required to complete the command.

Below are a few examples of creating tar files:

```
[student@stationX ~]$ tar cvf work.tar .bash_profile /tmp
...output omitted...
[student@stationX ~]$ tar cvf myHome.tar ~
...output omitted...
[student@stationX ~]$ tar cvf /dev/fd0 ~
...output omitted...
```

Copyright © 2006 Red Hat, Inc
All rights reserved

Inspecting Archives

- Syntax
  - **tar tf archive_name.tar**
  - **tar tvf archive_name.tar**
- First form displays a list of all files in the archive
- The **v** causes a long listing (like **ls -l**) of each file in the archive

21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat Inc
All rights reserved

Extracting an Archive

- Syntax
  - **tar xvf *archive_name.tar***
- Extracts to the current directory by default
  - Target can be specified with **-C** ***`dir`***
- Files maintain their hierarchy relative to the current directory

22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*"Un-tarring" **tar** archives*

Like archive creation and archive inspection, archive extraction can use the **v** option

tar extracts to the current directory by default, but you can specify another directory to change to before extracting with **-C** ***`/directory`***

There are many more options to **tar** to do things like preserve the original owner for each file and preserve the original permissions See the man page for details.

Why Use File Compression?

- Results in smaller file size
- Text files can be compressed over 75%
- Binary files usually don't compress much, if any
- **tar** archives are often compressed

23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The value of file compression

Files that are not used very often are often compressed. The advantage of a smaller file usually outweighs the added time it takes to compress/uncompress the file

While text files often have patterns that lead to compression ratios of over 75%, binary files rarely compress well with 0 - 25% being typical. In fact, it is possible for a "compressed" binary file to actually be larger than the original.

gzip – gnu zip

bzip2

Compression Utilities

- **gzip, gunzip**
  - Standard Linux compression utility
  - Over 75% compression for text files
- **bzip2, bunzip2**
  - Newer Linux compression utility
  - Generally achieves better compression than **gzip**

24

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

gzip archives

The standard Linux compression utility is **gzip** , a fast and efficient tool that is available on Linux, Unix, and non-Unix platforms. When a file `filename` is compressed with **gzip**, the compressed file is named *`filename`*.gz. The file can then be uncompressed using **gunzip** , recreating the original file `filename`

The **gunzip** command can also uncompress files compressed with the traditional Unix **compress** command, making **compress** essentially obsolete

bzip2 archives

A newer Linux compression utility is **bzip2**. Files compressed by this utility carry the extension `.bz2` and are uncompressed with **bunzip2** . **bzip2** compressed files are generally smaller in size than their gzipped counterparts.

Another compression utility available in Red Hat Enterprise Linux is **zip** This utility is compatible with the DOS/Windows PKzip/Winzip utilities and can compress more than one file into a single file, something **gzip** and **compress** cannot do It's useful to transfer file and directory archives to and from the DOS/Windows platform. **zip** archives are unpacked with **unzip**.

Linux and Unix users often prefer instead to use **tar** and **gzip** together in preference to **zip** This combination is often recognized by decompression utilities on other platforms, too. See the online documentation for more information on **zip**.

Using Compression

- Sample compression commands
  - **gzip termcap**
  - **gzip -v termcap**
  - **gunzip -c termcap.gz | wc -l**
  - **gunzip termcap.gz**

25

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Comparing compression tools

The compression commands take one or more file names as arguments. To uncompress the compressed files, similarly named commands are used, as shown above The **-v** option to the compression commands shows how much space was saved.

The following sequence shows the basic operation of these commands The `termcap` file is a text file that was copied from the `/etc` directory.

```
[student@stationX ~]$ ls -l termcap
-rw-r--r--    1 student  student    737535 Jan 17 15:41 termcap

[student@stationX ~]$  compress -v termcap
termcap:  -- replaced with termcap.Z Compression: 54.44%
[student@stationX ~]$  uncompress termcap

[student@stationX ~]$  gzip -v termcap
termcap:          67.7% -- replaced with termcap.gz
[student@stationX ~]$  gunzip termcap

[student@stationX ~]$  bzip2 -v termcap
termcap:  3.930:1,  2.036 bits/byte, 74.55% saved, 737535 in, 187668 out.
[student@stationX ~]$  bunzip2 termcap.bz2
```

The traditional **compress** command reduces the `termcap` file by more than half The standard **gzip** command does a better job, reducing the file to less than 1/3 the original size. Finally, the newer **bzip2** command reduces the file to 1/4 the original size.

The **-c** option to the **gzip** command leaves the original compressed file alone, but sends an uncompressed copy of the file to standard output. The **-d** option decompresses a file, making **gzip -d** `file.gz` equivalent to **gunzip** `file.gz`.

Copyright © 2006 Red Hat. Inc.
All rights reserved

Compressing Archives

- Often **tar** archives are compressed
- **tar** can compress/uncompress archives
- Compression switches - use during creation and extraction
  - **z** for **gzip** compression
  - **j** for **bzip2** compression

26

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*Combining **tar** with compression tools*

Compressing archive files that are mostly text-based is a common task. tar can perform the compression on the fly with either the **z** or the **j** option.

When creating gzipped tar archives, output files are often named with a `.tgz` extension rather than `.tar.gz`. (If **gunzip** is used on a `.tgz` file, it will leave the uncompressed archive with a `.tar` extension.)

What would the following commands do?

```
[student@stationX ~]$ tar czf smallHome.tgz ~
... output omitted...

[student@stationX ~]$ mv smallHome.tgz /tmp

[student@stationX ~]$ cd /tmp

[student@stationX ~]$ gzip -d smallHome.tgz
```

What is the resulting file name?

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 8

- Questions and Answers
- Summary
  - Linux filesystem structure
  - Using removable media
  - Using unformatted floppies
  - Archiving and compression

27

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Lab 8
The Linux Filesystem

Goal:	Develop a better understanding of Linux filesystem essentials including: the creation and use of links, using slocate and find, and archiving and compressing files.
Estimated Duration:	1 Hour
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 1: Creating and using links

Instructions:

1. Copy the file /usr/share/dict/words to your home directory:

```
[student@stationX ~]$ cd
[student@stationX ~]$ cp /usr/share/dict/words .
```

2. The /usr/share/dict/words file you just copied was actually a symbolic link. List the contents of /usr/share/dict to see the link and the file it references:

```
[student@stationX ~]$ ls -l /usr/share/dict
total 404
-rw-r--r--   1 root  root  409305 Feb 5  2003 linux.words
lrwxrwxrwx   1 root  root      11 Oct 3 17:33 words -> linux.wc
```

3. How can you tell that words is a symbolic link?

l in Long LIST du

4. Why is the file size field for words set to 11?

indicates PATH TO original file

5. The permissions string for words allows full access to everyone. What impact does this have on the linux.words file? Can users other than root use the link to write data to linux.words? USE SAME rights AS orig. FILE

NO

6. List the files again, this time displaying their corresponding inode numbers.

```
[student@stationX ~]$ ls -i /usr/share/dict
```

Do the two files have the same or different inode numbers?

NO

7. Now create both a symbolic and hard link in your home directory that point to the words file in your home directory:

```
[student@stationX ~]$ ln -s words soft
[student@stationX ~]$ ln words hard
```

Copyright © 2006 Red Hat Inc
All rights reserved

8. Test that your new links both function as pointers to the data in `words` (the head command prints the first 10 lines of a file):

```
[student@stationX ~]$ head hard soft
```

9. Examine the links that you have created with the commands that follow, then answer the questions below (the **stat** command presents inode information):

```
[student@stationX ~]$ ls -il hard soft
[student@stationX ~]$ stat hard soft
```

What is the size of `soft`? 5

What is the size of `hard`? 4992010

What is the link count for `hard`? 2

What is the link count for `soft`? 1

Who owns (UID/GID) `hard`? STUDENT/STUDENT

Who owns (UID/GID) `soft`? STUDENT/STUDENT

10. *Bonus Challenge*: If the instructor indicates that time permits, explore on your own to answer the following questions:

Can you make a symbolic link to a "target" that does not exist? Does the output of ls give you any indication of this condition?

Yes, FLASHES RED + WHITE.

Can you make a hard link to a target that does not exist? Why or why not?

No error, No such file/dir

Can you make a hard link to a symbolic link? What happens when you try?

No error, No such file/dir

After creating several hard links, is there any way to tell which is the "real" file? Is this even a valid question (in other words, is any file any more "real" than the hard links you created)?

____

Copyright © 2006 Red Hat Inc
All rights reserved

Sequence 2: Determining Disk Usage

Scenario: You want to document the amount of free space left on each of the filesystems on your system. Additionally, you want to have a list of which directories are consuming the most space on your system.

Instructions:

1. Use **df** to determine the amount of free space on each of your filesystems. Your output should resemble the following (although, depending on how your particular installation was performed, the output could vary).

```
[student@stationX ~]$ df
Filesystem           1k-blocks      Used Available Use% Mounted
/dev/hda5             12880648   1634344  10591988  14% /
/dev/hda1                36566      2476     32202   8% /boot
```

2. Note that the default operation of the df command is to report its information in blocks. Try using the **-h** and **-H** options, to report totals in "human readable" sizes instead:

```
[student@stationX ~]$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda5              12G  1.6G   10G  14% /
/dev/hda1              36M  2.5M   31M   8% /boot

[student@stationX ~]$ df -H
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda5              13G  1.7G   10G  14% /
/dev/hda1              37M  2.6M   32M   8% /boot
```

What is the difference between the two switches (Use man df)?

-h uses ~~[illegible]~~ division of 1024 while -H uses 1000

3. Use the **du** (disk usage) command from your home directory to determine how much space all of your files are consuming. Be sure to try the **-h** option for more readable output.

Copyright © 2006 Red Hat, Inc
All rights reserved

Sequence 3: Archiving and Compressing

Scenario: The primary hard drive on your system has started to make horrible noises every time you use it, and you suspect that it is about to die and take your valuable data with it. Since the last system backup was done 2 and a half years ago, you have decided to manually back up a few of your most critical files. The /tmp directory is stored on a partition on a different physical drive than the dying drive, so you will temporarily back up your files there. (However, since files in /tmp that have not been accessed for 10 or more days are deleted nightly, you'd better not store your critical data there for too long!)

Deliverable: Your "important data" safely archived, compressed, and backed up to the /tmp directory.

Instructions:

1. Store the contents of /etc in a tar archive in /tmp. The tar commands will output a few error messages because non-privileged users don't have read access to some files in /etc; for the purposes of this lab, they may be ignored:

```
[student@stationX ~]$ tar cvf /tmp/confbackup.tar /etc
....output omitted...
```

2. List the new file and record its size:

```
[student@stationX ~]$ ls -lh /tmp/confbackup.tar
....output omitted...
```

Size of your confbackup.tar file 36894720 (36M)

3. Use **gzip** to compress your archive. Then record the new file size:

```
[student@stationX ~]$ cd /tmp
[student@stationX ~]$ gzip -v confbackup.tar
[student@stationX ~]$ ls -lh confbackup.tar.gz
....output omitted...
```

Size of your confbackup.tar.gz file 7101706 (6.8M)

Size difference between compressed and uncompressed archive

29793014 (29,2M)

Copyright © 2006 Red Hat, Inc.
All rights reserved.

4. Uncompress the file, re-compress it with **bzip2**, and record the new file size:

```
[student@stationX ~]$ gunzip confbackup.tar.gz
[student@stationX ~]$ ls -lh confbackup.tar
...output omitted...

[student@stationX ~]$ bzip2 -v confbackup.tar
[student@stationX ~]$ ls -lh confbackup.tar.bz2
...output omitted...
```

Size of your confbackup.tar.bz2 file 5049677 (4,9M)

Size difference between compressed and uncompressed archive 31845043 (31,1M)

5. On a traditional UNIX system, the steps of archiving with **tar** and then compressing the archive would occur separately, much as you have done in the previous steps. On a Linux system with the GNU **tar** command, the tar-file can be filtered through a variety of compression programs automatically during the actual creation of the file. Try the following sequence of steps.

```
[student@stationX ~]$ rm confbackup.tar.bz2
[student@stationX ~]$ tar czf test1.tgz /etc
[student@stationX ~]$ tar cjf test2.tbz /etc
[student@stationX ~]$ file test*
test1.tgz:      gzip compressed data, from Unix
test2.tbz:      bzip2 compressed data, block size = 900k
```

Copyright © 2006 Red Hat, Inc
All rights reserved

Sequence 1 Solutions

3. There are two ways to tell that `words` is a symbolic link:

   - The first character in the file's mode is `l`, which denotes a symbolic link
   - The filename includes `-> linux.words`, which shows the link's target.

4. A symbolic link is really just a file that contains the name of another file. When a user accesses `words`, they are redirected to the file described by its contents: `linux.words`. Each character stored in a file takes up one byte There are 11 characters in "linux.words", therefore `words` contains 11 bytes worth of data.

5. Since `words` is just a pointer, it has no real permissions. Thus the `rwxrwxrwx` in `words`' mode is irrelevant since all requests to access `words` are redirected to `linux.words`, which has more restrictive permissions. In other words, non-root users would still not be able to alter `linux.words`, even if they referred to it as `words`.

6. Unlike hard links, which are multiple directory entries pointed at the same inode, symbolic links are separate files that contain a path (relative or absolute) to the intended target. Because symbolic links do point to paths instead of inodes, which are filesystem-specific, the link and its target do not have to be on the same filesystem

9. **What is the size of `soft`?** The size in bytes of a symbolic link is equal to the number of characters in the link's target. Since there are 5 characters in `words`, `soft` should have a size of 5.
   What is the size of `hard`? The size of a hard link should be equal to that of its target The size you see should be something like 4992010 but a different value is ok as long as it is the same for both `linux.words` and `hard`.
   What is the link count for `hard`? Since there are two files (`hard` and `words`) pointing to the same inode, the link count for that inode should be 2
   What is the link count for `soft`? Since `soft` is its own file with its own inode and there are no hard links pointing to that inode its link count should be 1.
   Who owns (UID/GID) `hard`? Although the original `/usr/share/dict/words.linux` is owned by the root user and the root group (UID 0/ GID 0), when you make a copy of a file the creator of the copy gets ownership by default. So if you were logged in as `student` when performing the copy, `~/words` should be owned by the `student` user and the `student` group, therefore so should any links to the same inode such as `hard`.
   Who owns (UID/GID) `soft`? Like copies, symbolic links are owned by their creators regardless of the target's ownership. Thus, `soft` should be owned by

Copyright © 2006 Red Hat, Inc
All rights reserved

whatever user you were logged in as when creating it as well as that user's primary group.

10 You can make a symbolic link to a non-existent target. Such links are called "broken" symbolic links and are displayed in flashing white text on a red background by **ls**.

Since a hard link refers to an inode, not just a filename, it is not possible to link to an inode if that inode does not actually exist.

You can create a hard link to a symbolic link. The hard link simply becomes another pointer to the symbolic link's inode. Thus, once created your hard link acts just like the symbolic link that it point to.

The original dentry for a file and a hard link pointing to the same inode are identical. They are both just names pointing to an inode. As such neither is more "real" than the other.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 2 Solutions

2. Both **-h** and **-H** use "human-readable" output. In other words, instead of saying that a filesystem's size is 102400 kilobytes, which is **df**'s default behavior, the human-readable version of its size would be listed as around 100M. The difference between the two arguments is that **-h** treats one kilobyte as equal to 1024 bytes, which is technically more accurate, and **-H** treats one kilobyte as equal to 1000 bytes in keeping with the real meaning of the "kilo" prefix. As a result they can deliver slightly different sizes. Remember that in both cases the size is likely to have been rounded to the nearest megabyte or gigabyte and so may not be exactly accurate.

Copyright © 2006 Red Hat Inc.
All rights reserved

Unit 9

Configuring the Bash Shell

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Be able to read and set shell variables
- Be able to export environment variables
- Know how to create aliases
- Understand how the shell parses a command line
- Know how to configure startup files
- Be able to use the **gnome-terminal**

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat Inc
All rights reserved

Configuring the Bash Shell

- The shell is configured through a variety of mechanisms:
  - Local variables
  - Aliases and functions
  - The **set** and **shopt** commands
- The shell can also configure other commands or applications through environment variables

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The **bash** shell can be configured through a number of mechanisms, including:

- local variables
- aliases
- functions
- the **set** command
- the **shopt** command

In addition, the shell can configure other commands and applications. Typically, this is done through a mechanism called *environment variables*.

All of these settings can be performed either at the system-wide level by the system administrator, or on a per account basis by the individual user.

In this unit, we will investigate how to configure the shell itself and how to use the shell to configure other programs.

Copyright © 2006 Red Hat Inc
All rights reserved

Variables

- A variable is a label that has a value
  - Used to configure the shell or other programs
  - Variables are resident in memory
  - Two types: local and environment
  - Local variables are used only by the shell
  - Environment variables are passed onto other commands
- Display variables and values using
  - **set** to display all variables
  - **env** to display environment variables

only

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The use of variables is central to the process of configuration, whether configuring the shell or some other program.

A variable is a label that equates to some value. The value can change over time, across systems, or across accounts, but the label remains constant. For example, a shell script may place a file in `$HOME`, a reference to the value of the variable `HOME`. This value may differ, depending on who is running the shell script, but the label `HOME` always accurately reflects the desired value.

Two types of variables exist: local variables, also called shell variables, and environment variables The difference between the two is that the shell will pass the environment variables on to commands that it calls, but it will not pass on local variables As a result, local variables are used to configure the shell itself, while environment variables are used to configure other commands.

The **set**, **env**, and **echo** commands can be used to display all variables, environment variables, and a single variable value, respectively. Examples:

```
[student@stationX ~]$  set | less
...output omitted...

[student@stationX ~]$  env | less
...output omitted...

[student@stationX ~]$  echo $HOME
/home/student
```

Copyright © 2006 Red Hat, Inc
All rights reserved

Common Local Variables

- `HISTFILESIZE` determines how many commands to be saved in the history file on logout
- `COLUMNS` sets the width of the terminal
- `LINES` sets the height of the terminal

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The `COLUMNS` and `LINES` variables are used for variable width terminals, such as xterm, gnome-terminal, or kterm, to establish the width and height of the terminal.

The history subsystem within the shell uses many variables to determine how much history to keep and how much history to save Among the useful history-related variables are:

`HISTFILE`

specifies the file in which history commands are stored on logout.

`HISTFILESIZE`

specifies the number of commands of history to be saved when the shell exits

`HISTSIZE`

specifies the number of history commands to keep while operating interactively

Copyright © 2006 Red Hat Inc
All rights reserved

The `PS1` Local Variable

- `PS1` sets the prompt
- Uses escape sequences to insert variable information in the prompt

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The `PS1` variable sets the prompt. The prompt can vary each time it is displayed by using special escaped sequences. These include:

\d	the date
\h	short hostname (not the fully qualified domain name)
\t	the current time
\u	user name (useful if you have multiple accounts)
\w	the current working directory
\!	the history number of the current command
\$	shows $ if you are a non-privileged user and a # if privileged user, useful if you sometimes become superuser.

Consider the following prompt setting:

```
[student@stationX ~]$ PS1="\u@\h:\w <\!>\$ "
digby@kennel:/tmp <1067>$
```

Note how the prompt has changed. The quotes are important as they encompass a concluding space to separate the prompt from the command that will be typed

For a complete list of these prompting escape sequences, see the PROMPTING section of the **bash** man page.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Aliases

- Aliases let you create shortcuts to commands

```
$ alias dir='ls -laF'
```

- Use **alias** by itself to see all set aliases
- Use **alias** followed by an alias name to see alias value

```
$ alias dir
alias dir='ls -laF'
```

alway used before commands

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Aliases

Aliases are shortcut names for longer commands If you have commands that you run often, but take a considerable amount of typing, you can reduce these to an alias:

```
[student@stationX ~]$ alias lf="ls -FCa"
```

The alias value must be a single word and so you will almost always want to quote the value as shown.

Even a relatively short command, if executed often, can save considerable typing if reduced to an alias:

```
[student@stationX ~]$ alias c=clear
```

Aliases can also be used for security purposes to force you to use certain flags:

```
[student@stationX ~]$ alias rm="rm -i"
```

In this case, if you ever want to use the **rm** command itself, instead of your alias, you can precede the command with a backslash:

```
[student@stationX ~]$ \rm -r Junk
```

Other Shell Configuration Methods

- Less common, but powerful commands to configure elements of the shell:
  - **set**
  - **shopt**

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The shell provides other configuration commands: **set** and **shopt**

The **set** command can configure many different aspects of the shell Examples:

set -b	Report termination of background commands immediately, rather than waiting for next prompt
set -u	Unsets variables generate an error
set -o noclobber	Prevents overwriting of files with > and >& operators
set -o vi	Enables vi syntax on **bash** command line instead of default emacs syntax

For a complete list of **set** values, see the **set** command under the SHELL BUILTIN COMMANDS section of the **bash** man page Also in this section is a list of items configurable through the **shopt** command.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Configuring Commands: Environment Variables

- Shell variables exist only in current shell instance
- Environment variables passed to *subshells*
- Shell variables can be *exported* into environment

```
$ EDITOR=/usr/bin/vim; export EDITOR
```

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Environment variables

Environment variables are set in the shell and used to configure other commands or applications. The shell passes on to all programs it calls the full list of environment variables; commands use these as they see fit For example, the EDITOR variable is used by a number of programs which need to invoke a text editor for the user. Setting this environment variable lets users use their favorite text editor instead of some default editor.

The traditional way to create an environment variable is to create a local variable and then export it. However, bash supports a condensed syntax for creating environment variables. Either of the following sequences will create an environment variable in bash:

```
[student@stationX ~]$ EDITOR=/usr/bin/vim
[student@stationX ~]$ export EDITOR
```

or:

```
[student@stationX ~]$ export EDITOR=/usr/bin/nano
```

Once exported, the value of the variable can be changed without needing to re-export the variable. To "blank" the value of an environment variable, use the **unset** builtin command:

```
[student@stationX ~]$ unset EDITOR
```

Copyright © 2006 Red Hat Inc
All rights reserved

Common Environment Variables

- HOME - Path to user's home directory
- LANG - Identification of default language rules to use
  - eg: en_US.UTF-8 for U.S. English
- PWD - User's current working directory
- EDITOR - Default editor programs should invoke for text editing
- LESS - Options to pass to the **less** command

OLDPWD = at Previous Working Direc

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Common environment variables and their meanings are shown above. The HOME and PWD variables are usually set for you. LANG is often set for you by the system administrator, but in a multi-lingual or multi-national environment, you may wish to override the choices of the administrator by resetting this variable

Editors can be very personal to users; often users prefer to set the EDITOR variable so that programs that respect this variable will bring up the editor of the user's choosing

The **less** command has many options; to force a set of options to always be used, set the LESS variable For example:

```
[student@stationX ~]$ LESS="-emqs"
```

Look up these options in the **less** man page

Other Standard Environment Variables

SHELL	Path to login shell
USER	Username of user
DISPLAY	X display name
VISUAL	Name of visual editor

Copyright © 2006 Red Hat, Inc.
All rights reserved

The TERM Environment Variable

- TERM environment variable sets the terminal type
- **reset** command (not variable) used to reset a terminal should the screen become corrupted

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

An important environment variable is the TERM variable, used by programs to determine your terminal type. If a terminal is improperly set, the computer will not be able to properly display data

When a terminal display becomes corrupted in some way (perhaps you accidentally viewed a binary file and as a result, an alternate font is set), run the **reset** command to remedy the problem

For more complex adjustments to your terminal settings, see the **stty** command. The **stty** command can perform a number of terminal settings to your system. Most ordinary users, even power users, will never use the **stty** command.

Copyright © 2006 Red Hat Inc.
All rights reserved

The PATH Environment variable

- The PATH environment variable is a colon separated list of locations where commands can be found
- **which** command showing location in the PATH of an executable

```
$ which xterm
/usr/bin/xterm
```

- Executable's location may be specified:

```
$ /bin/ls /etc
$ cd /bin; ./ls /etc
```

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The PATH environment variable contains a colon separated list of directories in which commands reside. When a command is executed and the path is not specified, then the shell will look in these directories in the given order, stopping on first match, to find the command Example:

```
[student@stationX ~]$  echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/digby/bin
```

The **which** command searches through the directories listed in the PATH environment variable looking for a matching executable When it finds one, it prints the path to the executable:

```
[student@stationX ~]$ which less
/usr/bin/less
```

It is possible to override the PATH variable by giving an explicit path to the command you wish to execute This may be either an absolute or relative pathname, but it must include a slash:

```
[student@stationX ~]$ ./less
```

This will run the **less** command in the current directory, instead of the **less** command that the PATH variable would find.

How bash Expands a Command Line

1. Split the line into words
2. Expand aliases
3. Expand curly-brace statements ({})
4. Expand tilde statements (~)
5. Expand variables ($)
6. Split the line into words again
7. Expand file globs (*, ?, [abc], etc)
8. Prepare I/O redirections (<, >)
9. Run the command!

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat. Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The process by which the shell interprets elements of the command line is extraordinarily complex Here, however, is a relatively simple version of the overall process:

1. The line is split into shell words, delimited by spaces, tabs, new lines, and a few other characters, and possibly overridden by single quotes, double quotes, and backslashes Tokens: space tab newline " ' | & ; ( ) < >

2. Function and alias expansion is performed. Whether a function or an alias takes precedence is a fairly complex matter.

3. Curly brace expansion, for such sequences as "cmd {o,c}", is performed. This may cause a change in word count, of course. Tokens: { , }

4. Tilde expansion is performed: ~, ~/, and ~username are all substituted for the appropriate strings. Token: ~

5. Parameter and variable substitution is performed. This includes arithmetic and command substitution In other words, all substitutions beginning with a $, plus the backquote characters If there are multiple substitutions, then changes will be made on the line from left to right. Common tokens: $ ${} $(()) $[] $() `

6. The line is split into shell words again.

7. File glob expansion is performed. Tokens: * ? [ ]

8. File redirection is performed. Common tokens: > >> < << 2> 2>>

Copyright © 2006 Red Hat Inc
All rights reserved.

9. The command is executed!

Copyright © 2006 Red Hat Inc.
All rights reserved.

Shell Startup Scripts

- Scripts of commands executed at login
- Uses include:
  - Configure the shell by setting local variables or running the **set** or **shopt** commands
  - Configure other programs through environment variables
  - Establish aliases
  - Run programs on startup

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Most of the configuration items that we have seen in this unit are only valid in the given shell. But typically, we will want settings to be established every time we start a shell, rather than having to type in all of our variables, aliases, and other commands on a per shell basis.

To accomplish this, we use startup scripts, scripts that run when a shell is created. The system administrator sets up some startup scripts, but individual users can control startup by editing scripts in their home directories

Copyright © 2006 Red Hat Inc
All rights reserved

Login Shells

- Login shells are first shells started (i.e. when you log in)
- Shells launched from a login shell typically are not login shells
- Login shells and non-login shells run different startup scripts

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

A critical concept to understand when it comes to startup scripts is the idea of the *login shell* A login shell is a shell that someone started by logging into the system. A non-login shell is a shell started up in some other way, perhaps by a user or a program issuing the **bash** command.

The login shell is an important concept because different startup scripts run, depending on whether a shell is a login shell or a non-login shell

Login - shell 5 scripts

non login shell 2 - scripts

Copyright © 2006 Red Hat Inc
All rights reserved.

Startup Scripts: Order of Execution

- Login shells
  - /etc/profile
  - /etc/profile.d
  - ~/.bash_profile
  - ~/.bashrc
  - /etc/bashrc
- Non-login shells
  - ~/.bashrc
  - /etc/bashrc
  - /etc/profile.d

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Login shells first call /etc/profile, which calls /etc/profile.d. Then, the file ~/.bash_profile is called. This file, in turn, calls ~/.bashrc, which calls /etc/bashrc. Each script in turn can undo changes made in previously called scripts. For example, the **PATH** variable is set in the /etc/profile script, but is then modified in the ~/.bash_profile script.

Non-login shells reference some of the same files, but note the difference in order. Non-login shells first call ~/.bashrc. This calls /etc/bashrc, which calls /etc/profile.d. Note that the /etc/bashrc file only calls /etc/profile.d for non-login shells; for login shells, the previously called /etc/profile calls the /etc/profile.d scripts.

Typical sorts of commands placed in startup scripts include:

local variable settings, particularly PS1

environment variable settings, such as PATH or LESS

aliases, or perhaps unaliases to remove undesired aliases set globally in earlier scripts

a umask, to be discussed in a later unit.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

/etc/profile

- System-wide startup script
- Parsed by all users with Bourne-style shells, including **bash** and **sh**
- Usually sets default `PATH` variable, user limits, and other variables and settings
- **bash** only sources `/etc/profile` if the shell is a login shell

17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The `/etc/profile` shell script is the first startup script run when a login shell is started It only runs for login shells; non-login shells do not invoke this script

The script will set a series of variables including `PATH`, `USER`, `LOGNAME`, `MAIL`, `HOSTNAME`, `HISTSIZE`, and `INPUTRC`.

It will also run scripts in the `/etc/profile.d` directory, discussed on the next slide

Copyright © 2006 Red Hat, Inc.
All rights reserved.

/etc/profile.d

- Some application-specific startup scripts in this directory
- Scripts called by a for-loop in /etc/profile
- Scripts set up variables and run initialization procedures

18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The /etc/profile.d directory contains additional setup shell scripts. Two copies of each script are listed in this directory, a Bourne shell style script with a .sh suffix, and a C shell style script with a .csh suffix. These do such things as set an alias for **ls** so that it displays colors by default.

The scripts in this directory are controlled by the system administrator; they cannot be modified by non-privileged users

Copyright © 2006 Red Hat, Inc.
All rights reserved

~/.bash_profile and ~/.bashrc

- For user-specific settings
- Common to place variable settings, aliases
- Commands that place output to the screen, such as the **date** command, should go in .bash_profile, not in .bashrc

19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The ~/.bash_profile and ~/.bashrc files are controlled by the individual user. The user may put variables, aliases, or any other startup command in these files. The ~/.bash_profile file is only called by login shells. The ~/.bashrc file is called by both login shells and non-login shells. Never put any command that may echo something to the screen in the ~/.bashrc file; such commands belong in the ~/.bash_profile file only.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

~/.bash_logout

- Resides in home directory
- Executed when exiting a login shell
- Useful for running programs automatically at logout
- Example uses:
  - Make backups of files
  - Delete temporary files
  - Display date and time of logout

20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The seldom used `~/.bash_logout` file runs when users log out Although most users do not use this file, it may be handy to do such housekeeping chores as making backup files, deleting temporary files or just displaying the logout date and time.

Copyright © 2006 Red Hat, Inc
All rights reserved

End of Unit 9

- Questions and Answers
- Summary
  - local and environment variables
  - command line parsing

21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc
All rights reserved

Lab 9
Configuring the bash Shell

Goal:	Develop additional skills in customizing the operation of the bash shell, including creating custom aliases
Estimated Duration:	45 minutes
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*

Sequence 1: Configuring the bash Shell

Deliverable: A system with a new aliases that clear the screen, and produce a useful time-sorted ls listing.

Instructions:

1. You've decided to create an alias so that when you type c, the system will run the clear command to clear the screen. Begin by logging in as user student on tty1, then input and test your alias.

```
[student@stationX ~]$ alias c='clear'
[student@stationX ~]$ alias
[student@stationX ~]$ c
```

2. This alias will be lost if you log out and log back in. To ensure that the new alias is available each time user student logs in, perform the following steps:

```
[student@stationX ~]$ cd
[student@stationX ~]$ vi .bashrc
```

   Locate the line that contains the text: # User specific aliases and functions Add your alias to the line immediately below:

```
alias c='clear'
```

   Save the file and exit.

3. Test your change by logging out, logging back in on `tty1`, and typing the following:

```
[student@stationX ~]$ alias
[student@stationX ~]$ c
```

Copyright © 2006 Red Hat, Inc
All rights reserved

4. Create an alias called **lr** that invokes **ls** with the following features:

- The alias displays a long listing format.
- It lists files beginning with a dot.
- It classifies files by appending a file type indicator to filenames of certain types.
- It sorts the listing by modification time.
- It displays files in reverse order.

You probably will need to consult the man page for the **ls** command to figure out the appropriate options. After you have created and tested your alias, add the alias to your `.bashrc` file.

Lr = Ls -altcr

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 2: Changing your bash prompt

Scenario: You've decided to customize your bash prompt to display the full path of the current working directory and the shell history number, and to make some other cosmetic changes.

Instructions:

1. In a terminal window, display the current value of your primary prompt string Note the presence of static ASCII characters and backslash-escaped special characters in the prompt.

```
[student@stationX ~]$ echo $PS1
```

2. Change your prompt to print only a static string for testing purposes:

```
[student@stationX ~]$ PS1='Red Hat Linux -> '
Red Hat Linux ->
```

Be sure to include a space after the -> so that commands have padding, that is, so that commands do not appear to touch the prompt (this is almost always desirable with prompts).

3. That's not too useful, so restore the traditional bash dollar-sign, along with the name of the host:

```
Red Hat Linux ->  PS1='\h $ '
stationX $
```

4. Insert the bash history prompt special character **\!** between the hostname and dollar-sign Make sure you pad it on each side with a space character. This will insert the number of the current command in bash's history The actual number you see will probably be different than the one shown below.

```
stationX $ PS1='\h \! $ '
stationX 21 $
```

5. Now refer to the **bash** man page to find the special character for the current working directory. (Search the man page for `PROMPTING`, and beware: you want the special character for the full pathname, not the basename as in the default prompt.) Insert that special character into your `PS1` prompt string preceded by a colon.

Copyright © 2006 Red Hat, Inc
All rights reserved.

6. Your customized prompt should appear as shown in the following examples. If not, continue to work on it.

```
station1:~ 21 $ cd /tmp
station1:/tmp 22 $
```

7. Edit your new definition of `PS1` into your `.bashrc`, then open a new terminal window to make sure your new prompt is in effect.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 3: Configuring Shell Options

Scenario: You will make several customizations to your **bash** shell environment using **set** and **shopt**.

Deliverable: A better understanding of various shell options.

Instructions:

1. Log in to `tty1` as user `student`. View a list of some of the currently configured shell options:

```
[student@stationX ~]$ set -o
allexport        off
braceexpand      on
emacs            on
errexit          off
hashall          on
....output truncated....
```

2. Note the current setting for the `ignoreeof` option (off). Verify its operation by pressing *Ctrl-d* to see if you are logged off.

3. Log back in to `tty1` as user `student` and execute the following to change, and then test, the `ignoreeof` option:

```
[student@stationX ~]$ set -o ignoreeof
[student@stationX ~]$ Ctrl-d
Use "logout" to leave the shell.
[student@stationX ~]$ set +o ignoreeof
[student@stationX ~]$ Ctrl-d
```

Copyright © 2006 Red Hat Inc
All rights reserved

4. You have now seen that when you try to execute a command from the shell prompt, you may in fact be running an external binary, a builtin shell command, an alias, a shell function, etc. Sometimes it may be important to determine exactly what the shell is running when you type a command. Use the builtin shell command **type** command to ask the shell what it is using to fulfill the following commands:

```
[student@stationX ~]$ type cat
cat is hashed (/bin/cat)
[student@stationX ~]$ type c
c is aliased to `clear'
[student@stationX ~]$ type set
set is a shell builtin
[student@stationX ~]$ type while
while is a shell keyword
```

Copyright © 2006 Red Hat Inc.
All rights reserved

Sequence 4: Command substitution

Instructions:

1 Determine the full pathname of the command executed when **metacity** is entered at the shell prompt. Use a shell shortcut to re-execute that command, appended with -message to run the same command on **metacity-message**, and then a second shell shortcut to run it on **metacity-window-demo**. Remember that *Esc-* in the instructions below means to press the Escape key.

```
[student@stationX ~]$ which metacity
[student@stationX ~]$ which Alt-.-message
[student@stationX ~]$ ^message^window-demo
```

2. Repeat your last command containing `ich`.

```
[student@stationX ~]$
Ctrl-richEnter
```

3. When a command contains another command enclosed in backquotes (``), `bash` evaluates the backquoted command first, and the result is substituted before the full command is executed

Use this technique to perform an **ls -l** listing on the full pathname of the command executed when nautilus is entered at the shell prompt Remember that which **nautilus** is evaluated first, its result is substituted on the command line, then **ls -l** is executed on the result.

```
[student@stationX ~]$ ls -l `which nautilus`
```

A dollar-sign followed by parentheses can be used to do the same thing as backquotes with the added advantage that parentheses can be nested. The following example uses **echo** and **date** to insert a datestamp before **ls -l**'s output. Note how the output of **whereis** is nested within the call to **ls**, which is in turn nested within the call to **echo**:

```
[student@stationX ~]$ echo "$(date): $(ls -l $(which nautilus))"
```

Unit 10

Advanced Topics in Users, Groups and Permissions

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc
All rights reserved

Objectives

Upon completion of this unit, you should:

- Know where Linux stores user, group and password information
- Understand how to change identities
- Know how to set default permissions
- Understand special permissions

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat Inc
All rights reserved.

User and Group ID Numbers

- User names map to user ID numbers
- Group names map to group ID numbers
- Data stored on the hard disk is stored numerically

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

User and Group IDs

When files are stored on the computer, the metadata about the file is stored numerically. That is, the user name and group affiliation of the file are not stored; rather, the user ID number and the group ID number are stored.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

/etc/passwd, /etc/shadow, and /etc/group files

- Authentication information is stored in plain text files:
  - /etc/passwd
  - /etc/shadow
  - /etc/group
  - /etc/gshadow

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

User and Group Information Files

When a user runs a command such as **ls -l** that displays user and group information about files, the numeric information is translated into names; it is the names that are displayed. The mappings of numbers to names are in the files `/etc/passwd` and `/etc/group`. The `/etc/shadow` file maps user names to their encrypted passwords and password and account expiration information. All files are colon separated.

The `/etc/passwd` file contains seven fields: user name, password placeholder (for historical reasons), uid number, gid number of the user's primary group, GECOS field (typically containing the user's real name), home directory, and shell to be run when a user logs in.

The `/etc/group` file contains four fields: group name, group password placeholder, gid number, and a comma separated list of group members.

The `/etc/shadow` file is referenced when someone logs in: the file contains a mapping of a user name to a password. For a complete list of the fields, see the man page:

man 5 shadow

Copyright © 2006 Red Hat, Inc.
All rights reserved

System Users and Groups

- Server programs such as web or print servers typically run as unprivileged users, not as root
  - Examples: `daemon, mail, lp, nobody`
- Running programs in this way limits the amount of damage any single program can do to the system

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

System Users and Groups

In addition to the ordinary user accounts and the `root` account for the superuser, a number of system users and groups exist. These accounts exist primarily so that server programs can run as non-privileged users or as particular groups.

System users and groups all have uid and gid numbers between 1 and 499. This excerpt from `/etc/passwd` shows several system users:

```
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```

Check the `/etc/group` file for sample system groups.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Changing Your Identity

- To change your password, run **passwd**
  - Insecure passwords are rejected
- To start a new shell as a different user:
  - **su**
  - **su -**
  - **su username**
  - **su - username**

No Login

? non login

Login shell

ctrl-d Log off

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Becoming another user

The **su** command is used to change identities. If a user name is not supplied, the **su** command assumes that you wish to become `root`. With the dash, **su** starts a login shell

Let's suppose bob has sue's permission to read her email He can temporarily assume sue's user id:

```
[student@stationX ~]$  whoami
bob
[student@stationX ~]$  su - sue
Password:
[student@stationX ~]$  whoami
sue
```

Since bob has now assumed sue's identity, he can now read sue's email and, in fact, has the same access to all sue's files that sue herself has. Note that this can only happen if bob knows sue's password.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

User Information Commands

- Find out who you are
  - **whoami**
- Find out what groups you belong to
  - **groups**, **id**
- Find out who is logged in
  - **users**, **who**, **w**
- Login/reboot history
  - **last**

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Getting User Information

The commands in the slide above provide information about users Experiment with the users, **who** and **w** commands Compare the output of these commands

Default Permissions

- Default permission for files is 666
- Default permission for directories is 777
- *umask* is subtracted from default to determine new file/directory permissions
- Non-privileged users' **umask** is 002
  - Files will have permissions of 664
  - Directories will have permissions of 775
- `root`'s **umask** is 022

deletes PERMISSIONS FROM USERS PERMISSIONS

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Default Permissions and umask

Without a umask in effect, any file created will have 666 permissions and any directory created will have 777 permissions. This means that anyone on the system will have read and write access to any newly-created file and similarly full permissions to any directories. In order to withhold permissions we use a umask. The umask lists the permissions to withhold. A umask of 002 will result in files created with 664 permissions and directories with permissions 775.

Note that execute privilege is always denied a newly-created file, regardless of the umask in effect. Execute privilege always has to be explicitly granted to a file Execute permission is given to a directory upon creation, unless explicitly denied by the umask.

The default umask on a Red Hat Enterprise Linux system is 002. To change your umask to 022, use the **umask** command:

```
[student@stationX ~]$ umask 022
```

umask is typically set by scripts run at login time. As a result, the next time you log in your umask will be set back to your default unless you add the command to one of your startup files, such as `.bashrc`.

Special Permissions

- Special permissions: a fourth permission set (in addition to user/group/other)
- Applicable in four cases:
  - suid for an executable
  - sgid for an executable
  - sgid for a directory
  - sticky bit for a directory
- Set with **chmod** or Nautilus

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Setting Special Permissions

In addition to the user, group and other permissions, an additional set of permissions exist called the special permissions. The special permissions are:

- The *suid*, or *set user ID*, bit
- The *sgid*, or *set group ID*, bit
- The *sticky bit*

The special permissions are rarely used and, in fact, are only effective in certain cases: the suid and sgid permissions are effective for executable regular files; the sticky bit and the sgid permission are effective for directories.

To set the special permissions, use the **chmod** command, preceding the usual three digits with a digit representing the special permission or permissions that you wish to have set: 4 for suid, 2 for sgid, 1 for the sticky bit. Example:

chmod 3775 groupdir

This would set both the sgid permission and the sticky bit for the directory groupdir.

You can also use Nautilus to set special permissions. To do this, right-click on a file or directory and choose Properties, then navigate to the Permissions tab. You will see a checkbox for each special permission under the Special Flags heading.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Special Permissions for Executables

- Special permissions for executables:
  - **suid**: command run with permissions of the *owner* of the command, not executor of the command
  - **sgid**: command runs with group affiliation of the group of the command

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The SetUID Permission

When the suid special permission is set for an executable, it means that the command will run with the authority of the owner of the file, rather than the authority of the user running the command. An example is the **passwd** command. The **passwd** command changes a user's password, which is stored in the `/etc/shadow` file and is not writable by non-privileged users. However, since the **passwd** command is owned by `root` and runs with the suid permission, users running the command have the privilege of `root` while changing their passwords. Hence, they have permission to edit `/etc/shadow`.

In a long listing, the suid permission is displayed as a lower case "s" where the "x" would otherwise be located for the user permission (an upper case "S" would be present if the underlying executable permission was not set):

```
[student@stationX ~]$  chmod 4551 passwd

[student@stationX ~]$  ls -l passwd
-r-s--x--x   1 root    root     15368 May 28  2002 passwd
```

The Set GID Permission

Similarly, commands running with the sgid permission run with the group affiliation of the group of the command:

```
[student@stationX ~]$  chmod 2551 same-gnome

[student@stationX ~]$  ls -l same-gnome
-r-xr-s--x   1 root    games    30855 Aug 13  2002 same-gnome
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

Special Permissions for Directories

- Special permissions for directories:
  - sticky bit: files in directories with the sticky bit set can only be removed by the owner and root, regardless of the write permissions of the directory
  - sgid: files created in directories with the sgid bit set have group affiliations of the group of the directory

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The Sticky Bit

The sticky bit for a directory sets a special restriction on deletion of files: with the sticky bit set, only the owner of the file, and the superuser, can delete files within the directory. An example of a directory with the sticky bit set is /tmp (the "t" denotes that the sticky bit is set):

```
[student@stationX ~]$  chmod 1777 /tmp

[student@stationX ~]$  ls -ld /tmp
drwxrwxrwt   30 root     root       4096 Mar  9 10:25 /tmp
```

The sgid permission for a directory means that files created in the directory will inherit its group affiliation from the directory, rather than inheriting it from the user. This is commonly used on group directories:

```
[student@stationX ~]$  chmod 2770 GroupDir

[student@stationX ~]$  ls -ld GroupDir

drwxrws---    2 digby    penguin   4096 Mar  9 10:35 GroupDir
```

Often both the sticky bit and the sgid permission will be set on a group directory:

```
[student@stationX ~]$  chmod 3770 GroupDir

[student@stationX ~]$  ls -ld GroupDir
drwxrws--T    2 digby    penguin   4096 Mar  9 10:35 GroupDir
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 10

- Questions and Answers
- Summary
  - User information is stored in /etc/passwd
  - Group information is stored in /etc/group
  - Special Permissions: Sticky Bit, SetUID, SetGID

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied. or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Lab 10
Switching Users and Setting a Umask

Goal:	Become familiar with the use of several essential commands in user identification and account switching.
Estimated Duration:	20 minutes
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 1: Local user logins

Instructions:

1. Log out of your workstation completely. Be sure you have exited from all virtual terminals and the X Window System.

2. Switch to virtual terminal 1 (`tty1`) by pressing: *Ctrl-Alt-F1*

3. Login into your workstation as user `visitor` with password *`password`*

   visitor

4. Determine information about this specific login by running the commands **whoami**, **groups** and **id**. Examine the output of these commands.

   visitor / uid=501(visitor), gid=502(visitor), groups 502, user_u:

5. Determine information about all concurrent logins on the workstation by running the commands **users**, **who** and **w** At this point, there should only be one user logged into the system The output of these commands will be more interesting as the sequence progresses.

6. Switch to virtual terminal 2 (`tty2`) by pressing: *Ctrl-Alt-F2*

7. Login into your workstation as user `student` with password *`student`*.

   501,501

8. Run the user information commands (**whoami**, **groups** and **id**) again and observe the output, noting how it differs from before.

9. Run the concurrent login commands (**users**, **who** and **w**) again and observe the output, noting how it differs from before.

(4) cont) system_r:unconfined_t

(5) users - visitor
who - visitor tty1

date

Copyright © 2006 Red Hat, Inc.
All rights reserved

Sequence 2: Switching user accounts

Instructions:

1 Switch to virtual terminal 1 (`tty1`) by pressing: *Ctrl-Alt-F1*

2. Record the output of the following commands:

id uid 501, gid=502 groups=502(visitor

pwd /home/visitor

3. Switch to the user `student` by running **su - student** and run the commands again:

id 500, 501, 501 (STUDENT

pwd /home/STUDENT

4. Run **exit** to terminate `student`'s login and return to your original `visitor` login.

5. Switch to the `student` account again, but this time run **su student** (without the hyphen). Run **id** and **pwd** again:

id 500, 501, 501 (STUDENT)

pwd /home/visitor

Why do these results differ from those you recorded in the previous step?

su - run a login shell while su does not

login script changes env to new user.

6. Log out of all the shells that you used during this sequence.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 3: Using umask to set default permissions on newly-created files

Instructions:

1 Log in to your workstation as `student`.

2. View your current umask

```
[student@stationX ~]$ umask
0002
```

3. Create a couple of files and a directory (do not examine the permissions yet).

```
[student@stationX ~]$ touch umtest1
[student@stationX ~]$ touch umtest2
[student@stationX ~]$ mkdir umtestdir1
```

4. Change your umask to a more paranoid (okay, maybe you prefer the word "secure") setting. Create new files and a directory.

```
[student@stationX ~]$ umask 027
[student@stationX ~]$ touch umtest3
[student@stationX ~]$ touch umtest4
[student@stationX ~]$ mkdir umtestdir2
```

Before looking at the permissions, what would you expect them to be?

umtest1 -rw-rw-r--
umtest2 -rw-rw-r--
umtestdir1 drwxrwxr-x
umtest3 -rw-r-----
umtest4 -rw-r-----
umtestdir2 drwxr-x---

List the files to see if you are correct:

```
[student@stationX ~]$ ls -ld um*
...output omitted...
```

Copyright © 2006 Red Hat, Inc
All rights reserved.

Sequence 4: Setting a Umask

Instructions:

1 Switch to virtual terminal 1 (`tty1`) by pressing: *Ctrl-Alt-F1*

2. Log in as the user `visitor` with a password of *`password`*.

3 Display your current umask:

```
[visitor@stationX ~]$ umask
```

0002

4. Below is a table of umasks. Fill in the table with the permissions of files and directories given the umask.

Umask	Directory Permissions	File Permissions
002	drwxrwxr-x	-rw-rw-r--
022	drwxr-xr-x	-rw-r--r--
007	drwxrwx---	-rw-rw----
027	drwxr-x---	-rw-r-----
077	drwx------	-rw-------

5. Decide on a reasonable umask for the `visitor` account. Set the umask at the end of the `.bashrc` file. Log out of the `visitor` account, log in again, and create a file and a directory. View the permissions. Did the directory and file permissions match your expectation? If not, revisit the table in step 4, above, and retry with a new umask.

027

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 2 Solutions

2 **id** `visitor`
pwd `/home/visitor`

3. **id** `student`
pwd `/home/student`

5. **id** `student`
pwd `/home/visitor`
The hyphen option to **su** initiates a new login shell, which includes changing the CWD to the new user's home directory and running the user's startup scripts (`~/.bash_profile`, etc). When **su** is run without the hyphen, your UID is changed, but all other details of the login session, including CWD and environment variables, remain the same.

Copyright © 2006 Red Hat, Inc
All rights reserved

Sequence 4 Solutions

4.

Umask	Directory Permissions	File Permissions
002	drwxrwxr-x	-rw-rw-r--
022	drwxr-xr-x	-rw-r--r--
007	drwxrwx---	-rw-rw----
027	drwxr-x---	-rw-r-----
077	drwx------	-rw-------

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Unit 11

Advanced vi/vim and Printing

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Be able to run text through a filter
- Understand search/replace syntax
- Understand advanced read/save operations
- Know how to configure **vi** and **vim**
- Be able to send text to the printer

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved

File Repositioning

CoPITAL LETTER

- G goes to last line in file
- 1G goes to first line in file (any number can be given and cursor will jump to that line)

(: LineNumber)

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*Moving in **vi/vim***

Often, it is useful to jump around in a file. The G command takes you to the bottom of the file. Precede the G command with a number and it will take you to that line number. A common G command is 1G, go to the first line. This is also useful when an error message tells you that an error exists on the particular line of a file. You can use the G command preceded by that number to jump right to the offending line.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Filtering

- The output of a command can be placed in the file
- The data in the file can be used as input
- Examples:
  - `!!date`
  - `!}sort`
  - `!}fmt -66`

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

A powerful feature of **vi** and **vim** is the ability to include into your document the output of Linux commands For example, if you want the date and time in your file, go to a blank line and, from command mode, issue the following command:

use filter & run Date

```
!!date
```

This will replace the current line with the output of the *date* command. Commands can also use the data in the document as input For example, if you have the following list in your file as a separate paragraph:

```
cow
aardvark
antelope
```

You can place your cursor on the first line and run the following command:

filter the next paragraph though sort command

```
!}sort
```

The `!` starts the filtering engine; the `}` passes into the sort command the paragraph below (remember the cursor movement meaning of `}`). The paragraph will then be replaced with the output of the sort command, hence sorting the paragraph. Another common command to use as a filter is the **fmt** command:

line wrapping

```
!}fmt -66
```

This will replace the paragraph with a paragraph formatted to be less than 66 characters wide. We will learn more about the sort and fmt commands in coming units

ex mode: Search and Replace

search /pattern/replace/where

- **sed** style search and replace
- Different default addressing rule
  - Changes current line only if address is omitted
  - `1,12`changes lines 1 through 12
  - `1,$` or `%` changes the entire file   % = 1,$ = whole file
  - `.,.+10` changes from current line (.) to current line plus 10 lines (`.+10`)
- Example: `:%s/Ohio/Iowa/g`

global

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

vi and **vim** can perform search and replace operations, much like the sed command. The primary difference between the sed and the other commands is that absent an address, sed works on the entire file, whereas with **vi** and **vim**, absent an address, they work only on the current line, the line on which the cursor resides. To make a change on the entire file, you must specify the lines:

:%s/Ohio/Iowa/g

In this command, the colon signifies that this is an ex command. `1,$` is the address, starting from line 1 and continuing through the last line. ;`s/Ohio/Iowa/`; indicates that for the string of characters `Ohio`;, replace the characters `Iowa`;. By default (and as with **sed**), only the first instance of `Ohio`; will be changed on any particular line. If you would like all instances of `Ohio`; on a line changed to `Iowa`;, then you put the trailing **g** character, as above.

The default substitution delimiter is the / character as seen above. However, vi treats whatever character follows the "**s**" command as the delimiter, so you can use other characters if necessary. This is especially useful in instances where the / character appears in your search or substitution strings. For example, to replace all instances of **/dev/hda** with **/dev/sda** you could do:

:%s/\/dev\/hda/\/dev\/sda/g

But note that the slashes in /dev/hda and /dev/sda had to be escaped to prevent vi from thinking they were delimiters. A much simpler option in this case would be to use a different delimiter:

:%s'/dev/hda'/dev/sda'g

shift : (enters EX mode)

use \ to escape

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Visual Mode

- Allows selection of blocks of text
  - *v* starts character-oriented highlighting
  - *V* starts line-oriented highlighting
  - *Ctrl-v* starts block-oriented highlighting
- Visual keys can be used in conjunction with movement keys:
  - w,), }, arrows, etc
- Highlighted text can be deleted, yanked, changed, filtered, search/replaced, etc.

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Visual Mode

It is possible to highlight characters, lines, or even block and then take actions on them (yank, delete, etc.). To begin visual mode, thus highlighting text, type:

- **v** to start character oriented visual mode
- **V** to start line oriented visual mode
- *Ctrl-v* to start block oriented visual mode

Move the cursor keys to highlight a section of text. Once the text is highlighted, type an action key and the action will be taken on the highlighted text. Action keys include:

- **c** to change selected text
- **d** to delete selected text
- **y** to yank (copy) selected text
- **gq** format to 'textwidth' columns

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Advanced Reading and Saving

ex mode (shift :)

- **:r newfile** inserts `newfile` contents into buffer
- **:r !date** inserts result of **date** into buffer
- **:1,20w xfile** writes lines 1-20 of buffer to `xfile`
- **:.,$w yfile** writes current line through end of buffer to `yfile`
- **:1,20w >> zfile** appends lines 1-20 to `zfile`
- **:n** switches to next open file
- **:n!** switches to previous open file
- **:n#** switches to open file #.

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

It is possible to read other files into your current file Use :r to read in a file:

:r newfile

This will read the file into the file, placing it after the cursor.

You can read in the output of a command by using *!commandname* after the :r. For example:

:r !date

Earlier in this unit, we learned to save the whole file. But it is also possible to save part of the file into another file A few examples with commentary:

:1,20w xfile	save line 1 through line 20 to a file called xfile
:.,$w yfile	save, starting from the current line through the end of the file, to a file called yfile
:16,40w >> zfile	append line 16 through line 40 to a file called zfile

If you are editing a file and wish to edit a different file, it is not necessary to exit **vi** or **vim** and then reenter. First, save the file, perhaps using **:w**, and then go to the next file with **:n**. For example: **:n otherfile**. If you do not wish to save your changes, you can go to the **otherfile**, abandoning changes, by running **:n! otherfile** and finally, to toggle between two files, call each up and then run **:n#** Each **:n#** will jump to the previous file, thus toggling between two files.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Using multiple "windows"

- Multiple documents can be viewed in a single **vim** screen.
  - *Ctrl-w, s* splits the screen horizontally
  - *Ctrl-w, v* splits the screen vertically
  - *Ctrl-w, `Arrow`* moves between windows
- Ex-mode instructions always affect the current window
- **`:help windows`** displays more window commands

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The **vim** editor provides many features beyond those of **vi**. Among the more interesting features is the ability of **vim** to display multiple buffers, multiple windows, simultaneously. To start **vim** in windowing mode, use the **-o** option: **vim -o** bashrc `.bash_profile`

This will open both files, showing bashrc in the top window and `.bash_profile` in the bottom window. Alternatively, if you open a file without the **-o** option, you can split an existing vim session, displaying the file in two windows by issuing *Ctrl-w, s* to split the screen horizontally or *Ctrl-w, v* to split the screen vertically

Initially, both windows will display whatever file you were viewing Changes made in one window will be merged into the other. Standard open commands such as **`:e`** ***`filename`*** can be used to change the file being edited in a window or you can run *Ctrl-w, n* to create a new window with a new, empty buffer in it.

The screen is split equally by default, but you can resize the current window with *Ctrl-w, +* to increase the size and *Ctrl-w, -* to decrease it.

Finally, you can jump from window to window by combining *Ctrl-w* with an arrow key This will move you to the next window in whichever direction the arrow would normally move your cursor. Instead of the arrow keys, you can also use **vi**'s traditional *h, j, k* and *l* keys for the same effect.

You can quit your current window with *Ctrl-w, q* or by using the Ex-mode **`:q`** command.

Extensive help on the vim windowing system is available with the command **`:help`** windows

Configuring vi and vim

- Configuring on the fly
  - `:set` or `:set all`
- Configuring permanently
  - `~/.vimrc` or `~/.exrc`
- A few common configuration items
  - `:set showmatch`
  - `:set autoindent`
  - `:set textwidth=65 (vim only)`
  - `:set wrapmargin=15`
  - `:set ignorecase`

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Dozens of configuration items exist for vi and vim. To examine your current configuration, run:

- **`:set`** lists a small number of important configuration items
- **`:set all`** lists the vast panoply of configuration items

To change a configuration item, use the **`:set`** command Some common items to set include:

- **`:set showmatch`** causes the cursor to momentarily jump to the matching left curly brace or left parenthesis when a right curly brace or a right parenthesis is typed. **`:se sm`** is an abbreviation of this. **`:set noshowmatch`** (**`:se nosm`**) turns off this behavior
- **`:set autoindent`** (**`:se ai`**) causes new lines to inherit the indentation level of the previous line. This is very useful for programmers. **`:set noautoindent`** (**`:se noai`**) turns this off.
- **`:set textwidth=65`** causes text to wrap (by inserting a hard return) when the text exceeds 65 characters (of course, any number can be given). **`:set textwidth=0`** turns this off. This option only works in **vim**.
- **`:set wrapmargin=15`** (**`:se wm=0`**) causes text to wrap when it reaches 15 characters from the right margin. **`:set wrapmargin=0`** turns this off.
- **`:set number`** (**`se nu`**) causes line numbers to be displayed in the left margin (visual only; line numbers are not actually stored in the document). **`:set nonumber`** (**`se nonu`**) turns this off.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

- **:set ignorecase** (**se ic**) causes searches to be case-insesnsitive (by default, they are case sensitive). **:set noignorecase** (**se noic**) turns this off.

- To save these settings so that they are run at every invocation of the editors, place the commands above in **~/.vimrc** The older **~/.exrc** will be read by both **vi** and **vim** if **~/.vimrc** does not exist.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Expanding your Vocabulary

- Learn more cursor movements
  - Expanding change, delete, yank, and put vocabulary
- Add the advanced material from the appendix to your skill base
- Learn more configuration features
- Play with filters!
- `:help`

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The **vi** and **vim** editors are tremendously featureful Over time, you may desire features that you have not yet mastered Many sources for additional information exist, including books and summary sheets. Some strategies for learning more include:

1 Learn more cursor movements As you expand your ready knowledge of cursor movements, you also expand your knowledge of change, delete, and yank commands, as most cursor movement characters can also be used with those commands Here is a partial list of useful cursor movements not discussed earlier

   - `0` moves to the start of the current line
   - `$` moves to the end of the current line
   - `^` moves to the first nonblank character of the current line
   - `e` moves to the end of the next word
   - `gg` moves top of the current file
   - *n*`%` moves to the line *n* percent through the current file
   - *n* moves to character *n* of the current line

2 Slowly, start to add elements from the Advanced section of this unit to your bag of tricks

3 Add in more configuration features, as needed.

4 Use filters! Experiment with filtering data, using data in the file as standard input.

5 Read the material in `:help` Learn to maneuver around the online help. Place the cursor over one of the `|tags|` and go to that tag with the Ctrl-] keystroke, returning to the previous screen with `:n#`. This is a rich resource well worth being mined extensively.

Copyright © 2006 Red Hat, Inc
All rights reserved

Printing in Linux

- Printers may be local or networked
- Print requests are sent to queues
- Queued jobs are sent to the printer on a first come first served basis
- Jobs may be canceled before or during printing

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Printing

Having created a file, you will no doubt want to print it. The printing system in Red Hat Linux is very flexible Printers may be parallel, serial, or networked Support is included for printing to remote CUPS IPP, lpd (common Linux and Unix printing subsystem), Windows, Netware, and JetDirect printers

Queues

One or more queues is associated with each printer. Print jobs are sent to a queue, not to a printer. Different queues for the same printer may have differing priority or output options Setting up print queues is the responsibility of the system administrator; individual users do not create print queues.

Jobs

Once a file has been sent to a queue for printing, it is called a job Jobs may be canceled while they are printing, or when they are in the queue waiting to be printed.

Copyright © 2006 Red Hat, Inc
All rights reserved

Printing Commands

- **lpr** sends a job to the queue to be printed
  - Accepts ASCII, PostScript, PDF, others
- **lpq** views the contents of the queue
- **lprm** removes a job from the queue
- System V printing commands such as **lp**, **lpstat** and **cancel** are also supported

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Using the print utilities

The **lpr** command is used to send a job to the printer. The Linux printing system will print files in ASCII, PostScript, PDF, and other formats. Most applications under Linux output PostScript format.

The **-P** option is used to select a queue other than the default and **-#** is used to specify the number of copies. For example, to print 5 copies of the file `report.ps` on the accounting printer:

```
[student@stationX ~]$ lpr -P accounting -#5 report.ps
```

When entered without options, **lpq** lists the jobs in the default queue. As with **lpr**, *-P* is used to specify a queue other than the default. For example:

```
[student@stationX ~]$ lpq

Printer: ps@localhost
Queue: no printable jobs in queue
Server: no server active
Status: job 'jay@localhost+916' removed at 12:16:03.083
Rank   Owner/ID                   Class Job Files        Size Time
done   jay@localhost+185             A  185 results      2067 08:38:04
```

To remove a job from the print queue, use **lprm** followed by the job number, specifying a non-default print queue if necessary. For example:

```
[student@stationX ~]$ lprm 916
Printer ps@localhost:
```

In this example, **lprm** responds with the name of the queue from which the job was removed. Note that a user may only remove his own print jobs from the queue.

Copyright © 2006 Red Hat, Inc
All rights reserved

Printing Utilities

- **ggv** views PostScript and PDF documents
- **xpdf** views PDF documents
- **enscript** and **a2ps** convert text to PostScript
- **ps2pdf** converts PostScript to PDF
- **lpstat -a** lists configured printers
- **mpage** prints multiple pages per sheet

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Tools to assist in printing tasks

Several utilities are included with Red Hat Linux to create output for the printer and interact with PostScript files.

enscript, a2ps

These commands convert text to PostScript and send it to the print queue or a file. They are often useful to send the output of a command to the printer via a pipe.

ggv

The ggv (GNOME Ghostview) utility is used to view PostScript and PDF files.

xpdf

The xpdf utility is used to view PDF files.

ps2pdf

This utility creates PDF files. You can use any program that will create a PostScript file, and then use ps2pdf to convert it to a PDF file. There are a number of versions of this program called **ps2pdf12**, **ps2pdf13**, and **ps2pdf14**, respectively creating PDF version 1.2, 1.3, and 1.4 output files.

pdf2ps

This utility coverts PDF files to PostScript, which makes it easy to print PDF documents right from the command line. There is also a pdftotext command which converts PDF documents to plain text documents.

mpage

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Prints ASCII or PostScript input with text reduced in size, so that multiple pages of input appear on a single sheet of paper.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 11

- Questions and Answers
- Summary
  - Move directly to line *x* with: *xG*
  - **!!*command*** displays output of *command*
  - **!}*command*** sends current line through *command*
  - **lpr** sends text to the printer

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Lab 11
Advanced Uses of the vi and vim Editors

Goal: Become familiar with more capabilities of the `vi` text editor

Estimated Duration: 1 hour

System Setup: A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Sequence 1: Advanced document management and manipulation

Instructions:

1 Retrieve the file `gpl.html` from server1 and store it in your home directory. The easiest way to do this is probably via ftp:

```
[student@stationX ~]$ cd
[student@stationX ~]$ lftp server1
lftp server1:~> get pub/gls/gpl.html
lftp server1:~> quit
```

You will use this HTML copy of the GNU General Public License for vi practice, and you will even change some of its text. Be assured that you are altering neither the letter nor the spirit of the GPL in this lab - nor will we be demonstrating good web design practice!

2. `vi` performs well when run in a terminal window - in fact, you'll probably run it in one most of the time. But for this lab, switch to a virtual console. Start `vi` on your copy of the file.

```
[student@stationX ~]$ vi gpl.html
```

Your cursor will appear on the first character of the first line of the file (a < symbol.) Note that the different types of HTML tags, keywords, values and other special characters are colorized by the `vim` extensions to `vi`.

Also take notice of the status line at the bottom of the terminal window. To the left you should see your cursor's current line and character number, which looks something like this: `1,1`.

3. In each task below, a series of keypresses is presented along with a description of what the effect of those keypresses should be. Work through the keypresses in each task and observe the results.

Troubleshooting tips:

If you do not observe the results described, return to the top of the file with ***Esc*** **`1G`** and begin the task again. (Note that the *Esc* keypress is only necessary if you are in insert mode.)

If you still do not observe the results described, you may have entered extraneous text into the file. If this happens, either exit vi with ***Esc*** **`q!`** ***Enter*** (again, *Esc* is necessary only when in insert mode) or re-start your vi session with **`:e!`**.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Keypress	New position of cursor
`l` or ***Right Arrow***	Right one character to the ! at 1,2
`5l` or `5`***Right Arrow***	The Y in DOCTYPE at 1,7 (movement commands can be prefixed with numerical "multipliers" - so, right 5 characters)
`j or` ***Down Arrow***	Down one line to the blank space at 2,7
`18j or 18`***Down Arrow***	Down 18 lines to the n in Everyone at 20,7
`h or` ***Left Arrow***	Left one character to the o in Everyone at 20,6
`4h or 4`***Left Arrow***	Left four characters to the v in Everyone at 20,2
`k or` ***Up Arrow***	Up one line to the 9 at 19,2
`w`	To the beginning of the next word - the I at 19,4 (moves to the next character-after-space)
`6w`	To the B at 19,29 (punctuation and groups of punctuation characters are treated as "words")
`b`	"Back" to the beginning of the last "word" - the comma at 19,27
`j3b`	Down one line, then back three word-beginnings to the p at 20,13
`e`	To the next end-of-word - the d at 20,21
`3e`	To the d at 20,33 (next end-of-word, repeated three times)
`5j)`	Down 5 lines, then to the beginning of the next sentence, at 26,34
`3)`	To the beginning of the third sentence after the starting point, at 32,51

Copyright © 2006 Red Hat, Inc.
All rights reserved

(	To the beginning of the previous sentence, at 31,12 - coincidentally, it begins with a left parenthesis
`3}`	To the beginning of the third paragraph after the current position, at 46,0. Paragraphs are delimited by blank lines; vi places the cursor at the beginning of the blank line before the third paragraph.
`2{`	To the beginning of the second paragraph before the current position, at 34,0

4. Return to the top of the document with **1G** before performing the next set of keypresses.

Keypress	New position of cursor
$	The last character on the current line, at 1,49
0	The first character on the current line, at 1,1
32| (32 followed by a pipe)	The 32nd character on the current line, at 1,32
G	The beginning of the last line in the file, at 381,1
77G	The beginning of the 77th line in the file, at 77,1
H	The first line in the current "screen", at 66,1
M	The middle line in the current "screen", at 77,1
L	The last line in the current "screen", at 88,1
Ctrl-b	Back one screen to 67,1, with cursor positioned on bottom line

Copyright © 2006 Red Hat, Inc.
All rights reserved

Ctrl-f	Forward one screen to 66,1, with cursor positioned on top line. vi provides an extra line of "context", which is why moving "forward" backs up one line in this case.
Ctrl-d	Down half a screen to 77,1, cursor positioned on same line
Ctrl-u	Up half a screen to 66,1, cursor positioned on same line

5. Searching for text not only helps find text to edit, but is also a handy way to position the cursor within the file. When you initiate a search command with either **/** or **?**, that character will appear on your status line. The pattern for which you wish to search will also be displayed as you type it. You must press *Enter* to actually initiate the search.

Return once again to the top of the document with `1G`, then try the next set of keypresses

Keypress	New position of cursor
`/GNU`	Search forward to beginning of GNU on fifth line of file
`nn`	Repeat last search twice - to beginning of GNU on ninth line of file. Repeated **`n`**s will move from one GNU to the next one until the search wraps around to the beginning of the file. (Don't try this yet.)
`N`	Repeat last search in the opposite direction - to beginning of GNU on eighth line of file.
`?General`	Search backward to General on fifth line.
`n`	Repeat last search to General on 375th line. (The search wrapped around the top of the file and re-started at the bottom.)
`NN`	Repeat last search, in the opposite direction, twice, to General on ninth line. (The search wrapped around the bottom of the file and re-started at the top.)

`/[Ff]r*ee`	Search forward for an upper- or lower-case f, followed by zero or more instances of r, followed by ee, to Free on line 18.
`n`	Repeat search, to freedom on line 26.
`11n`	Repeat search 11 times, to fee on line 48.
`''`	Return cursor to the line it was on before the last movement command. (This command is two single-quote marks.)

6. Next, try these insert-mode commands to enter new text into your document. Return again to the top of the file with **1G** before proceeding. Note that the examples below use the convention *Esc* to mean "press the escape key" and *Enter* to mean "press enter".

   Since you'll be changing HTML source, switch back to X and open `file:///home/student/gpl.html` in mozilla to review your changes. Continue using **vi** in your virtual console, and toggle to X when the tasks instruct you to view your changes.

Keypress	Explanation/Effect
18Gi<HR>*Esc*	Move down to line 18, enter insert mode, type <HR>, then leave insert mode. i enters insert mode and places subsequently typed text before the cursor position.
/License*Enter*ea (GPL)*Esc*	Search for the pattern License, move to the end of the word, enter insert mode, type a space, then (GPL), and leave insert mode. a enters insert mode and places subsequently typed text after the cursor position.
G2koLast updated: June 30, 1991*Esc*	Move to the end of the file, then up two lines. Enter insert mode (with a lower-case letter "o"), opening a new line below the current cursor position. Type the "Last updated" content, then leave insert mode.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

`1GO<!--edited by RH033 student-->`*Esc*	Move to the top of the file Enter insert mode (with an upper-case letter "O"), opening a new line above the current cursor position Type an HTML comment, then leave insert mode
`:w`*Enter*	Save your changes, but stay in vi.

Now use Mozilla's "reload" button to see how the browser rendered your changes. (The HTML comment will not display.)

7. Here are some text-changing commands to try. Return again to the top of the file with 1G before proceeding.

Keypress	Explanation/Effect
`/H1`*Enter*`RH2`*Esc*`n.`	Search for the next occurrence of H1, enter overstrike mode, type H2, leave overstrike mode, repeat the search, then repeat the last edit. You've just changed the HTML tags to display the header text between them in a smaller size. Save your change with :w, then re-display the document in mozilla to see the effect of this edit.
`/Boston`*Enter*`cwCambridge`*Esc*	Search for Boston, then "change word" it to Cambridge
`/1`*Enter*`r2`	Move to the next 1, and replace it with 2. Insert mode is not entered with r, so no *Esc* is required.
`/-`*Enter*`5x`	Move to the dash, and delete it and the last four digits of the nine-digit zip code.
`/U`*Enter*`dw`	Move to the next U, and delete the word under the cursor.
`/END OF TERMS`*Enter*`ddp`	Search for END OF TERMS, delete the line on which it appears, then paste it on the line below the cursor. This effectively switches the position of two lines.

Copyright © 2006 Red Hat, Inc.
All rights reserved

`kP`	Move up one line, then paste another copy of the deleted line above the current line. Deletes (and copies, demonstrated below) are placed in the "delete buffer". The contents of this buffer can be pasted as many times as desired.
`uuu`	Undo the last three changes.
`2`***Ctrl-r***	Redo the last two "undone" changes.
`1G/Also`***Enter***`kd}5jp`	Back to the top of the file; search for Also; move up one line; delete to the end of the next paragraph; move down five lines, and paste the contents of the delete buffer, effectively switching the order of the two paragraphs. Note that d can be followed by a movement command to delete to some specific location in the file.
`1G/general`***Enter***`dwelp`	Back to the top of the file; search for general; delete it, then move to the end of the next word, then one character to the right, and paste the deleted word. You have just switched the order of two adjacent words.
`bdwbP`	To the beginning of the pasted word; delete it, then back to the beginning of the previous word, and paste before the cursor. You have just replaced the words in their original order.
`1G/Version`***Enter***`2yyjp`	To the beginning of the file; search for Version; copy two lines, then move down one line and paste the copy under the current line.
`4}jy}}P`	Move fourth paragraphs down, then down one more line. Copy to the end of the paragraph, then move to the end of the paragraph and paste. You have just duplicated the paragraph.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

`/59`***Enter***`ywP`	Move to 59, then copy the word under the cursor and paste it before the cursor.
`1G:%s/H2/H4/g`***Enter***	To the beginning of the file; replace all occurrences of H2 with H4.
`1G:%s/[Ff]ree/FREE/g`***Enter***	To the beginning of the file; replace all occurrences of Free or free with FREE

Save your changes and quit with **`:wq`**, then re-load the document in mozilla to see the effect of the last two global substitution commands.

8. Repeat step one to re-retrieve `gpl.html` from server1 into your home directory, overwriting your edited copy. Start **vi** on the new copy, then try the following miscellaneous commands.

Keypress	Explanation/Effect
`27GJ`	Move to line 27, then join lines 27 and 28.
`$0`	Prove to yourself that you've created a long line by moving to the end of the line with $, then to the beginning of the line with 0.
`~~~~~~~~`	Change the case of the character under the cursor, then move to the right, seven times
`2k!}fmt -66`***Enter***	Up 2 lines, then format the current paragraph to a line length of 66 characters. The **!** command initiates a shell; the **}** specifies "from here to the next paragraph (it can be any movement command); and the command to run on the text specified by the movement command is **fmt -66**. The text returned by the shell replaces the original text.
`2>>`	Indent two lines by eight spaces.
`2j>}`	Move down past the two indented lines, and indent the rest of the paragraph by eight spaces.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

`186G!6jsort`*Enter*	Move to line 186, then filter the next six lines through **sort**.

9. This last set of tasks demonstrates some command-mode options that save and open files.

Keypress	Explanation/Effect
`:w gpledit.html`*Enter*	Write a copy of your current file to `gpledit.html`.
Ctrl-G	Show information about the file being edited - it is the original file, not `gpledit.html`.
`:e#`*Enter*	Switch to the last-edited file in this session. In this case, that's `gpledit.html`
`:35,41w gplextract.html`*Enter*	Write lines 35-41 of the current file to a new file, `gplextract.html`
`:57,63w>> gplextract.html`*Enter*	Append lines 57-63 of the current file to the existing file, `gplextract.html`.
`:e#`*Enter*	Switch to the last-edited file to have a look at `gplextract.html`.
`:e#`*Enter*	Now switch back to `gpledit.html`.
`1Gix`*Esc*	Move to the top of the file and insert an x at 1,1
`:e!`*Enter*	Discard edits in the current file, and re-load from the previously-saved copy.
`G:r gplextract.html`*Enter*	Move to the end of the file, then read in the contents of `gplextract.html` into the current file. The contents are placed below the current line.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

:q! *Enter*	Quit **vi**, abandoning your changes.

10 From the shell prompt, start **vi** on three files.

```
[student@stationX ~]$ vi gpl.html gpledit.html gplextract.html
```

This should start **vi** with the first file displayed. Perform the following commands to practice switching files.

Keypress	Explanation/Effect
:n *Enter*	Move to the next file in the list.
:n *Enter*	Move to the next file in the list.
:rew*Enter*	"rewind" to the first file in the list.
:wq! *Enter*	Save any pending edits and quit **vi** unconditionally

Copyright © 2006 Red Hat, Inc.
All rights reserved

Challenge Sequence 2: Learning more with vimtutor

Instructions:

1 If time remains in the lab, log in as user `student` on a virtual console (not a graphical terminal). Start **vimtutor** and follow the on-screen instructions until time runs out

```
[student@stationX ~]$ vimtutor
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Unit 12

Introduction to String Processing

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Know how to combine tools to perform complex string manipulations

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat .com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

head

- Displays first few lines (default: 10 lines) of text in a file

```
$ head /tmp/output.txt
```

- Use **-n** or **--lines** parameter to change number of lines displayed

```
$ head -n 20 /tmp/output.txt
```

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The head command is used to display just the first few lines of a file. The default is 10 lines.

```
[student@stationX ~]$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
```

-n specifies the number of lines to display:

```
[student@stationX ~]$ head -n 3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

tail

- Displays last few lines (default: 10 lines) of text in a file

```
$ tail /etc/passwd
```

- Use **-n** or **--lines** to change number of lines displayed

```
$ tail -n 5 /etc/passwd
```

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

tail is used to display the last few lines of a file The default is 10 tail is often used by the system administrator to read the most recent entries in log files.

```
[root@stationX ~]# tail /var/log/cron
root (10/13-08:20:00-659) CMD (/sbin/rmmod-as)
CRON (10/13-10:54:37-422) STARTUP (fork ok)
CRON (10/13-14:36:26-422) STARTUP (fork ok)
root (10/13-14:40:00-661) CMD (/sbin/rmmod-as)
CRON (10/13-17:53:08-422) STARTUP (fork ok)
root (10/13-18:00:00-656) CMD (/sbin/rmmod-as)
root (10/13-18:01:00-658) CMD (run-parts /etc/cron.hourly)
CRON (10/15-13:45:56-422) STARTUP (fork ok)
root (10/15-13:50:00-781) CMD (/sbin/rmmod-as)
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

tail
continued

- Use **-f** to **follow** the end of a text file as it changes
  - Used to "watch" log files

```
$ tail -f log.txt
```

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Using **-f** causes **tail** to continue to display the file in "real time", showing additions to the end of the file as they occur. This is very useful for watching growing files, such as the output of the make command System administrators use this feature to keep an eye on the system log using the following command:

```
[root@stationX ~]# tail -f /var/log/messages
```

tail -f will continue to show updates to the file until *Ctrl-c* is pressed.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

wc (word count)

- Counts words, lines, bytes and characters
- Can act upon a file or STDIN

```
$ wc story.txt
39      237     1901 story.txt
```

- Use **-l** for only line count
- Use **-w** for only word count
- Use **-c** for only byte count
- Use **-m** for character count (not displayed)

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

wc counts the number of lines, words, bytes and/or characters in a file. On traditional Unix systems every character in a text file took up exactly 1 byte, so the file size could be assumed to be equal to its size in bytes. However, with the advent of internationalization and larger character sets like Unicode some characters can take up to four bytes. Thus, if a document uses any non-ascii characters, the **-m** argument is required to get an accurate character-count.

If you specify more than one file, wc also reports totals for whatever values it has been told to count.

```
[student@stationX ~]$ wc .bash*
3       3       24  .bash_logout
13      29      191  .bash_profile
20      55      337 .bashrc
36      87      552 total
```

wc can also accept data on the standard input. This can be useful for counting the number of lines in a command's output. For example:

```
[student@stationX ~]$ ls -l /tmp | wc -l
32
```

shows us that **ls -l /tmp** produces 32 lines of output. Therefore there are 32 files and directories in `/tmp`.

sort

- Sorts text to stdout - original file unchanged

```
$ sort [options] file(s)
```

- Common options
  - **-r** performs a reverse (descending) sort
  - **-n** performs a numeric sort
  - **-f** ignores (folds) case of characters in strings
  - **-u** (unique) removes duplicate lines in output
  - **-t** *c* uses *c* as a field separator
  - **-k** *X* sorts by *c*-delimited field *X X*
  - **-k** *X,Y* sorts using field *X* followed by field *Y*

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

sort is used to sort text data This data can be in a file or the output of another command. **sort** is often used with pipes as in the example below which displays an alphabetical list of users whose login shell is set to **bash**:

```
[student@stationX ~]$ grep bash /etc/passwd | sort
alex:x:503:504::/home/alex:/bin/bash
gdm:x:42:42::/home/gdm:/bin/bash
joshua:x:500:500:Joshua M. Hoffman:/home/joshua:/bin/bash
root:x:0:0:root:/root:/bin/bash
star:x:504:505::/home/star:/bin/bash
```

The **-k** option sets the sort field. The following command will sort the /etc/passwd file by the uid number:

```
[student@stationX ~]$ sort -t : -k 3 -n /etc/passwd
```

The argument to the **-k** option can be two numbers separated by a dot. In this case, the number before the dot is the field number and the number after the dot is the character within that field with which to begin the sort.

The **-n** option sorts numerically, instead of by character. Without the **-n** option, the numbers 71, 12, and 9 would sort as 12, 71, and 9, whereas with the **-n** option, they will sort as 9, 12, and 71.

Copyright © 2006 Red Hat, Inc
All rights reserved

uniq

- Removes successive, duplicate lines in a file
- Can use in conjunction with **sort** to remove all duplicates
  - **sort -u** achieves the same effect
- Use **-c** to count number of occurrences of duplicate data

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

uniq "removes" duplicate adjacent lines from a file. To print only unique line occurrences in a file ("removing" all duplicate lines), input to uniq must first be sorted Since uniq can be given fields or columns on which to base its decisions, these are the fields or columns upon which its input must be sorted.

Used without switches, uniq removes duplicate lines in its input, using the entire record as a decision key

Use **-u** to output only the lines that are truly unique - only occurring once in the input.

Use **-d** to output only print one copy of the lines that are repeated in the input

Use **-c** to produce a frequency listing Each line will be prepended with a number indicating how many times it appears in the input

Use **-fn** or **-sn** to avoid comparing the first n fields or characters in each line, respectively.

The following example uses sort and uniq to list the shells used in `/etc/passwd:`

```
[student@stationX ~]$ cut -d: -f7 /etc/passwd | sort | uniq
/bin/bash
/bin/false
/bin/sync
/dev/null
/sbin/halt
/sbin/nologin
/sbin/shutdown
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

cut

- Display specific columns of file data
  - **$ cut -f4 results.dat**
  - **-f** specifies field or column
  - **-d** specifies field delimiter (default is TAB)
    - `$ cut -f3 -d: /etc/passwd`
  - **-c** cuts by characters
    - `$ cut -c2-5 /usr/share/dict/words`

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

cut is used to "cut" fields or columns of text from a file and display it to standard output

For example:

```
[student@stationX ~]$ cut -f3 -d: /etc/passwd
```

will display a list of uids from `/etc/passwd`, because uids are stored in field the third colon-delimited field delimiter.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Other String Processing Tools

- **paste** - paste files together
- **tr** - character translator

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

paste combines files "horizontally". It takes a line from each file and "pastes" them together to standard output, separated by a tab. Use the **-d** option to change the output delimiter:

```
[student@stationX ~]$ paste -d: ids.txt data.txt > merged.txt
```

tr is used to translate characters; that is, given two ranges of characters, any time a character in range 1 is found, it is translated into the equivalent character in range 2. This command is commonly used in shell scripts to ensure that data is in an expected case:

```
echo -n "Enter yes or no: "
read answer
answer="$(echo $answer | tr 'A-Z' 'a-z')"
```

In this example, the user is queried for data. If the user responds in lower case, the **tr** command will do nothing, but if the user responds in upper case, the characters will be changed to lower case.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Version Comparison with diff

- Compares two files for differences
- `[student@stationX ~]$ diff area.c /tmp/area.c`

```
33c33
<    x = y + 2;
---
>    x = y + 4;
```

- `33c33` indicates line where files differ
- < indicates line in first file
- > indicates line in second file

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

diff is used to compare the contents of two files for differences. If you upgrade a utility and would like to see how the new configuration files differ from the old, you would use diff. For example:

```
[student@stationX ~]$ diff /etc/named.conf.rpmnew /etc/named.conf

20c20
<      file "root.hints";
---
>      file "named.ca";
```

In the above example we can see that the line 20 of the first file reads `root.hints`, while line 20 of the second file read `named.ca` Other than that the files are identical.

Spell Checking with aspell

- Interactive spell-checker
- Easy way to check spelling in a file
  - *$ aspell check letter.txt*
- Can create personal dictionaries
- **look** - quick spell check

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

aspell is an interactive spell checker. It offers suggestions for corrections via a simple menu-driven interface.

```
[student@stationX ~]$ aspell check file.txt
Some times *peple* type stuff wrong.
1) people                    6) peel
2) Pele                      7) Pelee
3) Peale                     8) peopled
4) purple                    9) peoples
5) Peel                      0) pep
i) Ignore                    I) Ignore all
r) Replace                   R) Replace all
a) Add                       x) Exit
?
```

aspell -l will non-interactively list the misspelled words in a file read from standard input.

```
[student@stationX ~]$ aspell -l < standfast.txt
Carcrashes
Morningcharm
Braincheck
```

More information on aspell can be found at http://aspell.sourceforge net

A quick spelling dictionary lookup can be performed with the **look** command. It comes in handy when you need the spelling of a word of which you know the first few letters.

```
[student@stationX ~]$ look exer
exercise
exercised
exerciser
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

```
exercisers
exercises
exercising
exert
.... output truncated ...
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 12

- Questions and Answers
- Summary
  - Basic string processing
  - Simple regular expressions

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Lab 12
Introduction to String Processing

Goal:	Become familiar with several string processing utilities available on a Red Hat Linux system
Estimated Duration:	20 minutes
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Sequence 1: Basic String Processing

Deliverable: A numerically reverse-sorted list of the shells assigned to the user accounts on the machine

Instructions:

1. Make a copy of /etc/passwd in your home directory:

```
[student@stationX ~]$ cd
[student@stationX ~]$ cp /etc/passwd .
```

2. For every account on the system there will be a line in /etc/passwd. Using wc, count the number of lines in the passwd file.

```
[student@stationX ~]$ wc -l passwd
```

The number of accounts on your system 35

3. Generate a list of the various shells in use on the machine and place this in another file

```
[student@stationX ~]$ cut -d: -f7 passwd > shells
```

4. Look at the contents of your new shells file with cat. You will see that although the file contains information that is not organized in a very friendly way. Sort the lines of output and place the sorted data in a new file:

```
[student@stationX ~]$ sort shells > sorted.shells
```

5. Your file contains multiple occurrences of the same values. Use **uniq** to provide a count of how many times each value appears:

```
[student@stationX ~]$ uniq -c sorted.shells > uniq.sorted.shell
```

6. Why did you need to sort the output first before passing it through uniq? (Hint: try **uniq -c shells** and **man uniq**)

only matchs following lines

Copyright © 2006 Red Hat, Inc.
All rights reserved.

7. To provide a numerically reverse-sorted list of the shells in used on the machine (of course, the exact numbers on your machine may vary from those shown here):

```
[student@stationX ~]$ sort -nr uniq.sorted.shells
30    /sbin/nologin
6    /bin/bash
1    /sbin/shutdown
1    /sbin/halt
1    /bin/sync
1
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 2: Further exercises in string processing

Instructions:

1 Devise, then write down the solution to each exercise. Remember, the answer is the command you devise, not its output! The Answers are listed at the end of the lab, but you should try to work each task out on your own before checking the answers. Each answer should be a single command line, and should implement at least one pipe.

2 The **aspell** command has no man page; how else might you find help?

aspell -?

3. How many files are in the directory `/usr/bin`? The output should be a single integer. Hint: devise a command that lists the filenames one per line, then think about how you can count those lines.

ls -1 /usr/bin/ | wc -l (1797)

4. List the misspelled words in the `/usr/share/doc/nautilus-*/NEWS` file.

aspell -l < /usr/share/doc/nautilus-*/NEWS

5 How many unique lines are in the output of the previous step?

103

aspell -l </usr/share/doc/nautilus-*/NEWS | sort | uniq | wc -l

Copyright © 2006 Red Hat, Inc
All rights reserved

Sequence 1 Solutions

6. You need to sort data before sending it to **uniq** because it only removes adjacent duplicates. Sorting puts all the duplicates together so that **uniq** can deal with them properly.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Sequence 2 Solutions

2. `aspell --help | less`

3. `ls -1 /usr/bin | wc -l`

4. `aspell -l < /usr/share/doc/nautilus-*/NEWS`

5. `aspell -l < /usr/share/doc/nautilus-*/NEWS | sort | uniq | wc -l`

Copyright © 2006 Red Hat, Inc
All rights reserved

Unit 13

String Processing with Regular Expressions

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Objectives

Upon completion of this unit, you should:

- Understand the importance of regular expressions
- Know how to use **grep**
- Know how to use **sed**
- Know how to use **less**

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Pattern Matching with Regular Expressions

- Regular expressions are a pattern matching engine
- Used by many tools, including: **grep**, **sed**, **less**, **vi**, **awk**
- Values:
  - Power over ease of use
  - Greed!

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Regular expressions, abbreviated *regex*, are a pattern matching system used by many Linux tools. Unlike the shell's file name generation pattern matching system, regular expressions are designed for power over ease of use. Indeed, regular expressions can look intimidating at times, although with a bit of practice and careful parsing, they should be comprehensible to an experienced Linux user

Among the tools that use regular expressions are:

- **grep** analyzes the contents of files a line at a time, returning lines that match a pattern;
- **sed**, the stream *ed*itor, returns the contents of a file (or stream of data), performing the specified search and replace instruction;
- **less**, which uses regular expressions in search commands;
- **vi**, which uses regular expressions for searches (like less) or search and replace (like sed);
- **awk**, a data oriented programming language

Regular expressions are *greedy* That is, if a regular expression can match a smaller string or a bigger string, it will always match the largest string possible

Regular expressions can be divided into three broad categories: wildcard characters that stand for some other character; modifiers, that modify the preceding character; and anchors that anchor a character sequence to the beginning or end of a line or the beginning or end of a word.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Wildcard Characters

- Wildcard characters stand for another single character:
  - `.` any single character
  - `[abc]` any single character in the set
  - `[a-c]` any single character in the range
  - `[^abc]` any single character *not* in the set
  - `[^a-c]` any single character *not* in the range

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Wildcard characters in regular expressions represent some other character. The valid wildcard sequences are:

`.` (a single period) represents any single character

`[abc]` represents any single character in the set, that is, either "a" or "b" or "c"

`[a-c]` represents any single character in the range. Some common ranges include: [a-z], [0-9], [aeiou], although any random range is permitted, such as [Digby9]. Note that this does not stand for the word "Digby9" but rather for any *one* of those six characters.

`[^abc]` or `[^a-c]` are similar to the structures above, but represent any single character *not* in the set or range.

Imagine a hit-and-run accident, in which witnesses disagreed on the license plate number. One says that it was 9HBE368; another, 9HEE398, and another 9NEB348. The police find that none of these are exactly correct. A regular expression could be used to search a database of license plates using these witness statements:

```
9[NH][EB][EB]3[469]8
```

Copyright © 2006 Red Hat, Inc
All rights reserved.

Character Classes

- Pre-defined character ranges
- `[[:`*`keyword`*`:]]`
- `[^[:`*`keyword`*`:]]` (negated)
- Keywords include
  - **alpha**, **upper**, **lower**, **digit**, **alnum**, **punct**, **space**
- Easier and more reliable than `[ ]` ranges for some tasks

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Simple character ranges like `[a-z]` can be useful, but they can also cause unexpected problems. Suppose you wanted to match any lower-case letter. The character range `[a-z]` would seem to be an appropriate choice for doing this. Indeed, on a Red Hat Enterprise Linux 4 system, it would work as expected and match all lower-case letters. However, on other operating systems, including Red Hat Enterprise Linux 3 and systems using a language other than English, you may find that `[a-z]` matches most upper-case letters as well. This is because of differences in the way that characters are ordered in different operating systems. On most operating systems, alphabetical characters are sorted with the lower-case letters before the upper-case letters, like this: a,b,c, ...z,A,B,C...Z. But on some systems and in some languages, the sorting order mixes cases: aAbBcC...zZ. On one of these systems the character range `[a-z]` would match every letter, regardless of case, except for the upper-case 'Z', which follows 'z'.

In short, the problem with simple character ranges is that their behavior can differ between systems. Character classes, however, provide predefined ranges that always behave the same way. Character classes are written as: `[[:`*`keyword`*`:]]` where *keyword* represents the type of characters you want to match. Supported keywords include:

lower	Lower-case letters	[a-z]
upper	Uppers-case letters	[A-Z]
alpha	All letters	[a-zA-Z]
digit	All numbers	[0-9]
alnum	All letters and numbers	[0-9a-zA-Z]
space	All whitespace (tabs, spaces, etc)	
punct	All punctuation	

A character class can be negated by adding a carat inside the outer braces. For example, `[^[:alnum:]]` would match any non-alphanumeric character.

Character classes have another advantage over character ranges in that they are easy to remember. One only has to type the name of the class instead of a list of its member characters. Consider having to type `[-~`@#$%^&*()_-+={\[}\]|\\;:'",<.>/?]` instead of just `[[:punct:]]`!

Modifiers

- Modifiers determine the number of the previous character
  - `*` zero or more of the previous char
  - `\+` one or more of the previous char
  - `\?` zero or one of the previous char
  - `\{i\}` exactly i of the previous character
  - `\{i,\}` i or more of the previous char
  - `\{i,j\}` i to j of the previous character

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Modifiers modify the previous character or the previous expression. Of the modifiers above, the single most difficult one to understand is the asterisk, *. The asterisk stands for zero or more of the **`previous`** character. Examples:

`a*` represents zero or more "a"s

`ab*c` represents an "a" followed by zero or more "b"s followe dby a "c". Note that this would match the strings "abbc", "abc", and even "ac Can you understan dwhy the last item matches?

`[[:alpha:]][0-9]*[[:alpha:]]` represents two letters of the alphabet separated by zero or more numbers

The rather cumbersome curly braces preceded by backslashes act as counters They specify the number of the previous character required. Note the differences in meaning depending on whether the curly braces contain a single number, a number followed by a comma, or two numbers separated by a comma. Examples:

`r\{6\}` represents exactly six "r"s

`#\{3,5\}` represents three, four, or five pound signs, "#" (aka octothorpes)

`[0-9]\{9,\}` represents nine or more numbers (not necessarily the same nine numbers)

The `\+` and `\?` sequences also modify the previous character as described in the slide above. Be warned that this syntax may not be portable across all Unix-like operating systems.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Anchors

- Anchors match the beginning or end of a line or word
  - ^ the beginning of a line
  - $ the end of a line
  - \< the beginning of a word
  - \> the end of a word

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The tools that use regular expressions process one line at a time. Therefore, regular expression characters exist to anchor an expression to the beginning and end of lines Examples:

`^Hi` the string "Hi" begins the line Note that a line beginning "Highway" would be matched

`^Hello$` lines contain only the characters "Hello"

Because string processing is often word bound, the regular expression system provides word anchors. The sequence \< matches the beginning of a word and the sequence \> matches the end of a word Absent the preceding backslashes, the < and > characters are ordinary characters with no special meaning. Examples:

`\<cat` represents any word beginning with "cat": cat, catalog, category, for example

`cat\>` represents any word ending with "cat": cat, polecat, Muscat, for example

`\<cat\>` represents the word cat

`<cat>` represents the word "cat" enclosed in angled brackets (here the angled brackets are ordinary characters)

The | Operator

- Represents a logical OR operation
- Must be escaped with \
- Used in larger regexes with \(and \)
  - `John\|Jane`
  - `Name: \(John\|Jane\) Smith`

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

A character range or class allows one to specify multiple characters, such as using **[ab]** to match 'a' or 'b'. But what if you wanted to match strings instead of characters? This can be done with the logical OR operator, \|, as in **'John\|Jane'**, which matches 'John' or 'Jane'

OR'd statements can be embedded within a larger regular expression by using \(and \) as in the following example: **'Name: \(John\|Jane\) Smith'**

This would match 'Name: John Smith' or 'Name: Jane Smith'

Copyright © 2006 Red Hat, Inc.
All rights reserved.

regex Combinations

- Regular expressions are most useful in combination with each other
  - `.*` zero or more of any character
  - `\*` a literal asterisk
  - `\<cat\>` the word 'cat'
  - `ab..ef` ab and ef separated by two chars
  - `.\{32\}` 32 of any character
  - `[[:alpha:]]*` zero or more letters

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Regular expressions can be most useful in combination with each other. Perhaps the single most common and useful regex combination is .*, which stands for zero or more of any character. Indeed, the period is commonly used with other regular expression characters, as demonstrated in the examples above.

When an asterisk follows a bracketed sequence, such as in the **`[[:alpha:]]*`** example above, it matches as any one of the characters in the brackets, not just the first one repetitively.

Matching an exact number of any character can be done with repeated periods, as in the example **`ab..ef`** above, but matching a large number of unknown characters is best done by using the period with a counter. In the example on the slide above, it is unlikely that the typist will be able to count out 32 periods accurately, and certainly later, such a sequence would be unsupportable (how many periods? how many should there be?) but the use of the single period with the counter is relatively easy to implement and, importantly, self-documenting (to the cognoscente).

A special character in regular expressions preceded by a backslash, \, is said to be quoted and so has no special meaning, but rather stands for itself only. For example, `\*` represents a literal asterisk. Obviously, this is not true for modifiers and anchors which require the backslash to be significant.

Copyright © 2006 Red Hat, Inc
All rights reserved

Regular Expressions - Examples

- What do the following match?

1. `Sm.th`
2. `Sm[iy]th`
3. `www\.redhat\.com`
4. `^#!`
5. `\<the`
6. `^[[:lower:][0-9] ]\{28\}$`
7. `[jJ]\(oe\|ane\)`
8. `^^Yipes!$$`

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

1. Smith, Smyth, Sm)th, Sm th, ...
2. Smith and Smyth
3. www.redhat.com
4. #! (at the beginning of the line)
5. Any word beginning with "the". This will include words such as "theory".
6. A line containing exactly 28 lower-case letters of the alphabet, numbers, or spaces.
7. Any of the strings Joe, joe, Jane or jane
8. ^Yipes!$ as the entire line. The first ^ and the last $ mean the match is anchored at both the beginning and end of the line. The second ^ is literal, as is the word Yipes, the exclamation point, and the first $. Thus, only ^Yipes!$ on a line by itself will be found.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Quote your regex's!

- On the command line, quote regular expressions
- File name generation characters must remain unquoted
- Do not use quotes in regular expressions within commands

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

When using regular expressions on the command line, it is critical that you quote the pattern, single quotes preferred. The shell parses the command line. When it sees characters such as the asterisk, square brackets, and backslashes, it will attempt to do file name expansion or other interpretation on the characters. Often, this has no effect, as the pattern will match no filename. However, it is possible that the pattern will match a file name and so corrupt your regular expression. For this reason, it is vital that you quote your regular expressions, and, of course, equally vital that you do not quote file name generation characters. Example:

```
[student@stationX ~]$ grep 'mail.*tar' *.txt
```

This will search all files with filenames ending in .txt for lines containing the words mail and tar separated by zero or more characters. However, if the single quotes are left out, and the current directory contains two files called `mail.old.tar` and `mail.star`, then this is how the shell will construct the command line:

Commands using regular expressions outside the command line do not require quotes. For example, using regular expressions to search for text while in the **less** command do not require quotes.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

grep

- Prints lines of files where a pattern is matched

```
$ grep john /etc/passwd
john:x:500:500:John Doe:/home/john:/bin/bash
```

- Also used as filter in pipelines

```
$ ls | grep .c
```

- Uses regular expressions

```
$ grep '[0-9][[:upper:]]\{3\}[0-9]\{3\}' cars
```

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

grep displays the lines in a file that match a pattern It can also process standard input. The pattern may contain regular expression metacharacters and so it is considered good practice to always quote your regular expressions. Examples:

To list lines containing either "Cat" or "cat" from the `pets` file

```
[student@stationX ~]$ grep '[Cc]at' pets
```

To list only lines of output from **ps**, which lists running processes, that contain the string "init": from the ps command

```
[student@stationX ~]$ ps ax | grep 'init'
```

Common **grep** options include:

-v	return lines that do *not* contain the pattern
-n	precede returned lines with line numbers
-c	only return a count of lines with the matching pattern
-l	only return the names of files that have at least one line containing the pattern
-r	perform a recursive search of files, starting with the named directory
-i	perform a case-insensitive search
-E	Use "extended" regular expressions, which more closely resemble the way regular expressions are handled in mos tprogramming languages. The most notable difference betwee nstandard and extended regular expressions is that extende dregular expressions do not require a backslash for operator slike |, { and }.

Copyright © 2006 Red Hat, Inc.
All rights reserved

sed

- **stream editor**
- Reads a file or stream of data; writes out the data, performing search and replace instruction(s)
- Uses regular expressions in search string (but not replace string)

Search & Replace

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The **sed** command is the stream editor, used to perform edits on a stream of data. Given a file name to process, sed will perform a search and replace on all lines in the file, sending the modified data to standard output; that is, it does not actually modify the existing file. As with grep, sed is often used in pipelines. A basic **sed** command may look like this:

```
[student@stationX ~]$ sed 's/cat/dog/' pets
```

In this example, the pets file will be sent to standard output with the string "cat" being replaced by the string "dog". By default, sed makes a maximum of one change per line. To instruct sed to make multiple changes per line, the "g" command, standing for global, should be appended to the end of the search and replace pattern:

```
[student@stationX ~]$ sed 's/cat/dog/g' pets
```

changes ALL matches in all lines

The search portion of the search and replace pattern is a regular expression, although the replace portion is not. For example, if "cat" may appear with an initial capitalization, the following command may be effective:

```
[student@stationX ~]$ sed 's/[Cc]at/dog/g' pets
```

every time it occurs

As sed works on strings by default, and not by words, word anchors may be useful:

```
[student@stationX ~]$ sed 's/\<[Cc]at\>/dog/g' pets
```

Although the replacement string is not a regular expression, it does have its own special syntax. Particularly useful is the "&" character, which stands for "whatever you found in the search portion". For example:

```
[student@stationX ~]$ sed 's/\<[Cc]at\>/& and dog/g' pets
```

This will replace the word cat, regardless of initial case, with "cat and dog", preserving case. Cool!

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Using sed

- Quote search and replace instructions!
- **sed** addresses
  - **sed 's/dog/cat/g' pets**
  - **sed '1,50s/dog/cat/g' pets**
  - **sed '/digby/,/duncan/s/dog/cat/g' pets**
- Multiple **sed** instructions
  - **sed -e 's/dog/cat/' -e 's/hi/lo/' pets**
  - **sed -f myedits pets**

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

As with **grep**, it is good practice to quote **sed**'s search and replace string, as demonstrated in the examples below.

By default, **sed** operates on all lines in a file It is possible to provide sed with addresses limiting replacements to just those lines. For example:

```
[student@stationX ~]$ sed '10,35s/cat/dog/' pets
```

In this example, the entire pets file will be sent to standard output, but the replacement of "cat" for "dog" will only be performed on lines 10 through 35, inclusive.

A fancy but seldom used feature of sed is that addressing can be done by string searches. For example:

```
[student@stationX ~]$ sed '/digby/,/duncan/s/cat/dog/' pets
```

In this example, starting on the line that contains the string "digby" and continuing through the line that contains "duncan", the string substitution of "dog" for "cat" will be performed on the pets file As before, the entire file will be sent to standard output, but changes will be performed only on the specified lines.

It is common to make several changes on a file Two different methods are provided by **sed** to perform multiple edits. For a small number of edits, the **-e** option indicates that a search and replace pattern is forthcoming, and can be used several times on a command line:

```
[student@stationX ~]$ sed -e 's/cat/dog/g' -e 's/cow/goat/g' pets
```

For a larger number of edits, or to save edits for the future, place them in a file and use **-f** to invoke them:

```
[student@stationX ~]$ sed -f myedits pets
```

Copyright © 2006 Red Hat, Inc
All rights reserved.

less and vi

- Searches in **less** and **vi** use regular expressions
  - **/h[aeiou]t**

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The search functions in the **less** and **vi** use regular expressions. This can be particularly useful if you are searching for a word regardless of case:

```
/[Cc]at
```

or if you are searching for a particular pattern, such as a date and time in a file:

```
/Mar  2 14:[0-2][0-9]
```

This will search for a date and time in the file between 2:00pm and 2:30pm on March 2nd

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 13

- Questions and Answers
- Summary
  - . matches any character
  - + matches one or more of the preceding
  - `[abc]` matches a range of characters
  - `{x,y}` matches between x and y of the preceding

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Lab 13
String Processing with Regular Expressions

Goal:	Become familiar with several string processing utilities available on a Red Hat Linux system.
Estimated Duration:	45 minutes
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*

Sequence 1: String processing with grep

Instructions:

1. Use **grep** to display the line for any account that starts with the letter g in your home directory's copy of /etc/passwd:

```
[student@stationX ~]$ grep '^g' passwd
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
gdm:x:42:42::/var/gdm:/sbin/nologin
```

   In this example, as in others below, your output may vary.

2. Display the line for any account that is using the **bash** shell:

```
[student@stationX ~]$ grep 'bash$' passwd
root:x:0:0:root:/root:/bin/bash
student:x:500:500::/home/student:/bin/bash
visitor:x:501:501::/home/visitor:/bin/bash
```

3. Display the line for any account that is not using the bash shell:

```
[student@stationX ~]$ grep -v 'bash$' passwd
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
...output truncated...
```

4. In order to illustrate the use of **diff**, create a modified copy of passwd. Start by using **grep** to remove any lines that contain the letters "N" or "P":

```
[student@stationX ~]$ grep -v '[NP]' passwd > modified.passwd
```

5. For a final change, use **tr** to convert any remaining capital letters to lowercase:

```
[student@stationX ~]$ tr "A-Z" "a-z" < ↙
modified.passwd > modified2.passwd
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

6. Use **cat** to view both your original passwd and the new modified2.passwd. Without looking very carefully, the differences are not easy to spot, even in this small file. Imagine if your passwd file contained entries for thousands of users! Use **diff** to produce a list of the changes between the two files:

```
[student@stationX ~]$ diff modified2.passwd passwd
14a15,16
> ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
> nobody:x:99:99:Nobody:/:/sbin/nologin
16a19,23
> nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
> sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:...
> rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
> rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
> nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:...
20c27
< xfs:x:43:43:x font server:/etc/x11/fs:/sbin/nologin
---
> xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
```

Again, your output may vary

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 2: Regular Expressions and String Processing

Scenario: Where there are blank lines below tasks, devise, then write down the solution. Remember, the answer is the command you devise, not its output! The Answers are listed at the end of the lab, but you should try to work each task out on your own before checking the answers. You might need to refer to man pages.

Instructions:

1. Try using **grep** to output only those lines of `/usr/share/dict/words` that contain a text pattern. For example, display the lines that contain the pattern `fish`:

```
[student@stationX ~]$ grep fish /usr/share/dict/words
blowfish
bluefish
codfish
...output truncated...
unselfish
unselfishly
unselfishness
```

2. Using the man page for the **grep** command, construct and then test a command that will output each line containing the pattern `fish`, and the two lines immediately before and after the matching line (to provide additional context).

   grep -B1 -A1 'fish' words >tempfish

3. Using the man page for grep, construct and then test a command that will output just a count of the number of lines on which the pattern `fish` appears in the words file.

   805 grep -c 'fish' words

4. Using the man page for `grep`, construct and then test a command that will output a line for each occurrence of the pattern `fish` within the words file, including the line number on which the match was made.

   grep -n 'fish' words > tempfish2

5. List the words in `/usr/share/dict/words` that contain `t`, a vowel, then `sh`.

   grep 't[aeiou]sh' words > tsh

Copyright © 2006 Red Hat, Inc
All rights reserved.

6. Create a regular expression that matches the words abominable, abominate, abomine, anomie, anomite, atomise, and atomize (but no others) in /usr/share/dict/words.

grep '^a[bt]omi[nestz][ae][bt]\?[le]\?[e]\?$' words > temp words

7. How many words in /usr/share/dict/words contain t, a vowel, then sh at the end of the word? Construct, then execute a command that produces just the count.

grep -c 't[aeiou]sh' words (418)

8. List the words in /usr/share/dict/words that contain exactly sixteen letters.

grep '^.\{16\}$' words

9. A useful source of information (not to mention plain text files) are the subdirectories of /usr/share/doc. We'll use files in the subdirectory provided by bash for the next few tasks. Since the directory name includes bash's version number, which may have changed if updates have been installed, you will use tab-completion when specifying the search location.

List the names of the regular files in /usr/share/doc/bash-*Tab* that contain the word expansion (in the files themselves, not the filenames).

grep -l expansion /usr/share/doc/bash-3.0/* (-F)

10. Produce a count of the number of times the pattern Linux appears within each regular file in /usr/share/doc/bash-*Tab*, but don't print a count for those files where the pattern appears 0 times. Hint: produce a count for all files, look at the output, then think about how you can suppress lines matching a certain pattern from the output.

grep -c 'Linux' /usr/share/doc/bash-3.0* | grep -v ':0'

11. List the filenames of all files below /usr/share/doc that contain the pattern Havoc.

grep -R -l 'Havoc'

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 3: Stream editing with regular expressions

Scenario: Consider a file called 'cats' containing the following list of words:

```
cat
catalog
concatenate
polecat
Cat
```

For each **sed** command below,enter the new text for each line of the file. Enter 'No Change' if the line would be unaffected.

Instructions:

1. **sed 's/cat/dog/' cats**

   cat ___dog___
   catalog ___dogalog___
   concatenate ___condogenat___
   polecat ___pole dog___
   Cat ___cat___ (No Change.

2. **sed 's/[Cc]at/dog/' cats**

   cat ___dog___
   catalog ___dog alog___
   concatenate ___condogenate___
   polecat ___pole dog___
   Cat ___dog___

3. **sed 's/\<[Cc]at/dog/' cats**

   cat ___dog___
   catalog ___dogalog___
   concatenate ___No change___
   polecat ___No change___
   Cat ___dog___

4. **sed 's/[Cc]at\>/dog/' cats**

   cat ___dog___
   catalog ___~~dog~~ No___
   concatenate ___No___
   polecat ___pole dog___
   Cat ___dog___

Copyright © 2006 Red Hat, Inc
All rights reserved

5. **sed 's/\<[Cc]at\>/dog/' cats**

cat *dog*
catalog *No*
concatenate *No*
polecat *No*
Cat *dog*

6. **sed 's/\<[Cc]at\>/& and dog/' cats**

cat *cat and dog*
catalog *No*
concatenate *No*
polecat *NO*
Cat *Cat and dog*

7. Create the 'cats' file Run the sed commands, above, and test your answers

Copyright © 2006 Red Hat, Inc
All rights reserved

Sequence 2 Solutions

2 `grep -B2 -A2 "fish" /usr/share/dict/words`

3 `grep -c "fish" /usr/share/dict/words`

4 `grep -n "fish" /usr/share/dict/words`

5. `grep "t[aeiou]sh" /usr/share/dict/words`

6. `"^a.omi.*e$"`

or:

`"\<a.omi.*e\>"`

The trick for this task is to determine what the words have in common and what is variable. Note that the number of characters in the words is one of the variables and so the `.*` syntax is required.

7. grep "t[aeiou]sh" /usr/share/dict/words

8. `grep "^................$" /usr/share/dict/words`

or:

`grep "^.\{16\}$" /usr/share/dict/words`

9 `grep -l expansion /usr/share/doc/bash-`***Tab***`/*`

10 `grep -c "Linux" /usr/share/doc/bash-`***Tab***`/* | grep -v ":0"`

11 `grep -R -l "Havoc" /usr/share/doc`

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 3 Solutions

1. **cat** dog
   catalog dogalog
   concatenate condogenate
   polecat poledog
   Cat No Change

2. **cat** dog
   catalog dogalog
   concatenate condogenate
   polecat poledog
   Cat dog

3. **cat** dog
   catalog dogalog
   concatenate No Change
   polecat No Change
   Cat dog

4. **cat** dog
   catalog No Change
   concatenate No Change
   polecat poledog
   Cat dog

5. **cat** dog
   catalog No Change
   concatenate No Change
   polecat No Change
   Cat dog

6 **cat** cat and dog
   catalog No Change
   concatenate No Change
   polecat No Change
   Cat Cat and dog

Copyright © 2006 Red Hat, Inc.
All rights reserved

Unit 14

Finding and Processing Files

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Be able to use **slocate**
- Be able to use **find**
- Be able to use **the Gnome Search tool**

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

slocate

- Can be invoked as **slocate** or **locate**
- Queries a pre-built database of paths to files on the system
  - Database must be updated by administrator
  - Full path is searched, not just filename
- May only search directories where the user has read and execute permission

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

slocate is the "Security Enhanced Version of Locate" On your Red Hat Enterprise Linux machine you can use the commands **slocate** or **locate** to query the slocate database of files. If you look at the `/usr/bin/locate` file you will see that it is really a symbolic link to the **slocate** command.

slocate or **locate** are commands that query a database of files looking for files that correspond to search criteria. This database must be generated by an administrator running the **updatedb** command Since an out-of-date database can be worse than useless, database updates can be also automated by an administrator enabling the `DAILY_UPDATE` option in `/etc/updatedb.conf`.

The slocate database only stores file name and path information so the **locate** and **slocate** commands only do searches based on file name. However, **locate** and **slocate** can use regular expressions when searching this database to provide information to perform a "fuzzy" search

slocate Examples

- **slocate foo**
  - Search for files with "foo" in the name or path
- **slocate -r '\.foo$'**
  - Regex search for files ending in ".foo"
- Useful options
  - **-i** performs a case-insensitive search
  - **-n** *x* lists only the first *x* matches

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Using an exact string as your **slocate** argument will query the database and return all files that have the target string in their names. However, realize that when a database entry is made for a file, the filename is stored as the absolute path. Which means that if you were to perform a search like **slocate images**; not only will you get files that are named images, or have images in their name, but also any named my_family_images or boot images, but also any file that has images on it's directory path such as

`/usr/share/backgrounds/images/space` or
`/usr/share/gimp/2.0/help/images/dialogs/dialogs-icon-floating.png`

When using regular expressions with **slocate**, remember to quote the regular expression as many regular expressions use shell meta characters. Also, **slocate** only accepts basic regular expressions not extended regular expressions.

If you decide to use **-n** to limit the number of results printed by **slocate, slocate** will only print the first -n <num> results.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

find

- **find [directory...][criteria...]**
- Searches directory trees in real-time
  - Slower but more accurate than **slocate**
  - CWD is used if no starting directory given
  - All files are matched if no criteria given
- Can execute commands on found files
- May only search directories where the user has read and execute permission

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Unlike **slocate**, **find** will do a real time search of the machines file system to find files that match the criteria of the command line arguments But realize that since **find** is looking at files in the file system as your user account, you must have read and execute permission on a directory to examine its content

find requires an argument of what directory it should start finding files in. So if you only wanted to find in a user student's home directory you would give **find** a starting directory of `/home/student`. If you would like **find** to look over the entire system, you would provide a starting directory of `/`.

find has a huge amount of options that can be provided to describe exactly what kind of file should be found. You can search based on file name, file size, last modified time stamp, inode number, and many, many more.

One of the features of **find** is that with the **-exec** and **-ok** options, which will be described in more depth on later pages, you can execute commands on the files that **find** has reported. Thus, **find** allows you to perform arbitrary actions on arbitrary files Very powerful.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Basic *find* Examples

- **find -name snow.png**
  - Search for files named snow.png
- **find -iname snow.png**
  - Case-insensitive search for files named snow.png, Snow.png, SNOW.PNG, etc
- **find -user joe -group joe**
  - Search for files owned by the user *joe* and the group *joe*

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Finding files based on their name will, unlike **slocate,** look for files that are named the exact string passed to the **find** command. That is to say, if a **find** command was used like:

```
[student@stationX ~]$ find / -name .png
```

findwould only return the files that were named **.png**, not files that contained in their name the string **.png**. Fortunately, you can use shell wild cards with **find**, but they must be quoted. As an example:

```
[student@stationX ~]$ find / -name "*.png"
```

would find all files on the system that have **.png** as the end of their name. Related to **-name** is the **-iname** option which works the same way as **-name** but performs a case-insensitive search.

The **-regex** option in **find** does not work quite the way one would expect **-regex** applies the regular expression to the name of the file, including the absolute path to the file. So in the above example, the regular expression **".*W.*\.png"** will match all files that have a capital **W** and **.png** in their names. If we had instead used **"W.*\.png"**, we would get no results because the full path to our file name will always begin with a / and our provided regular expression indicates we are only interested in files whose full path name begins with a **W**, something that can never be true

Copyright © 2006 Red Hat, Inc
All rights reserved.

find and Logical Operators

- Criteria are ANDed together by default.
- Can be OR'd or negated with **-o** and **-not**
- Parentheses can be used to determine logic order, but must be escaped in bash.
  - **find -user joe -not -group joe**
  - **find -user joe -o -user jane**
  - **find -not \(-user joe -o -user jane \)**

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The **find** command by default will logically and all file parameters together. For example:

```
[student@stationX ~]$find / -name "*.png" -user student -mtime +12
```

will find files whose name ends in **.png** and are owned by the user student and have a last modified time stamp greater than 12 days ago. But we could logically or these parameters together:

```
[student@stationX ~]$find / -name "*.png" -o -user student -o -mtime +12
```

Now **find** will print the names of files that end in **.png** or are owned by the user student or were modified more than 12 days ago.

find also includes a logical not operator. Options preceded with a **!** or the **-not** option will cause the criteria **find** is looking for to be opposite. To illustrate:

```
[student@stationX ~]$find / -name "*.png" -not -user student
```

This will find files whose name ends in **.png**, but that are not owned by the student user account.

With the addition of logical operators, the question of operator precedence is raised. Logical ands have a higher priority than a logical or, and a logical not has a higher priority than an and or an or. To force precedence of an expression, you can enclose options that should be grouped together in parentheses. Make sure to put spaces after the \(and before the \) or find will not work as expected. The find man page has more details about this in the OPERATORS section.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Executing Commands with find

- Commands can be executed on found files
  - Command must be preceded with **-exec** or **-ok**
    - **-ok** prompts before acting on each file
  - Command must end with ***Space*\;**
  - Can use {} as a filename placeholder
  - **find -size +102400k -ok gzip {} \;**

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Using the **-exec** or **-ok** options with find will cause **find** to execute a command once for each file that matches the given criteria This is commonly used for things like removing files or renaming files to have a certain extension Be extremely careful when using **-exec** because it may perform the action on many files (remember that find recurses through subdirectories) and it does not ask for confirmation Remember that running the search without **-exec** will list all matches, thus allowing you to preview which files will be acted upon Alternately you can use the **-ok** option, which causes **find** to ask for each file

The reason that the commands given with **-exec** and **-ok** must end in a **\;** is because find uses **;** as the delimiting character. Unfortunately **;** is also a delimiting character for the shell so we must prevent bash from interpreting it. When a character is prepended with a backslash (\), **bash** is instructed to treat it literally, so typing **\;** at the **bash** command prompt will send ; to **find** after **bash** has done its interpretation.

Copyright © 2006 Red Hat, Inc
All rights reserved

find Execution Examples

- **find -name "*.conf" -exec cp {} {}.orig \;**
  - Back up configuration files, adding a `.orig` extension
- **find /tmp -ctime +3 -user joe -ok rm {} \;**
  - Prompt to remove Joe's tmp files that are over 3 days old
- **find ~ -perm +o+w -exec chmod o-w {} \;**
  - Fix other-writable files in your home directory

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

You can use **find** to execute any command you would like, and if you use the {} as filename place holders, the **find** command will put the file name of a file it has found in place of {} and execute the command
If your **-exec** command does not include {}, **find** will still execute the command once for each file that is found

The Gnome Search Tool

- Actions->Search for Files...
- Graphical tool for searching by
  - name
  - content
  - owner/group
  - size
  - modification time

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The Gnome Desktop software includes a graphical file finding agent. This program uses the find command in the back-ground, but does not have all of finds features implemented

To execute this utility, click the Actions button on the top panel and choose Search for Files... from the menu.

By default, this utility will only look at the user's home directory, but it can be modified to also look on the system's file system or removable media devices such as floppies and CD-ROMs.

Under the Show more options switch, you can customize your search to include items like the file last modification time, file size, file owner, and others.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 14

- Questions and Answers
- Summary
  - Use **slocate** to quickly find files that are not new
  - Use **find** to search based on very specific criteria and optionally run commands on matching files
  - Use the Gnome Search Tool for an intuitive, but powerful GUI search tool.

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Lab 14
Finding and Processing Files

Goal: Develop a better understanding of the **find** tool

Estimated Duration: 1 Hour

System Setup: A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 1: Using find

Scenario: Log in as user `student`. Devise and execute a **find** command that produces the result described in each of the following problems, then write down the command in the space provided.

You may need to refer to the man page for find. Remember that you can search **man** pages with ***/string***.

You will encounter a number of `Permission denied` messages when find attempts to recurse into directories to which you do not have read access - do not be concerned about these messages. Better yet, you can suppress these error messages by appending your find command with **2> /dev/null** (more about this in the next unit.)

Instructions:

1 Produce a listing of everything under `/var/lib` owned by user `rpm`.

find /var/lib -user 'rpm'

2: Produce a listing of everything under `/var` owned by user `root` and group `mail`.

find /var -user root -group mail

3. Produce an **ls -l** style listing of everything on the system not owned by users `root`, `bin` or `student`

This will take a long time, so run the command in one terminal and do the next problem in another terminal while it runs.

find / -not -user root -not -user bin -not -user student -

4. Produce an **ls -l** style listing of everything under `/usr/bin` with size greater than 50 kilobytes

find /usr/bin -size +50k -ls

5 Execute the **file** command on everything under `/etc/mail`

find /etc/mail -exec file {} \;

Copyright © 2006 Red Hat Inc.
All rights reserved

6. Produce an **ls -l** style listing of all symbolic links under `/etc`

Hint: **man find** and search for an option that searches for files by type

find /etc -type l

7. Produce an **ls -l** style listing of all "regular" files under `/tmp` that are owned by user `student`, and whose modification time is greater than 120 minutes ago

Hint: **man find** and search for an option that searches by modification time in minutes

find /tmp -type f -mmin +120 -user student

8. Produce a listing of all executables under `/bin` and `/usr/bin` that have the SetUID bit set

find /bin ~~&~~ /usr/bin -perm -u+s

9. Modify the command above to find all "regular" files under `/tmp` that are owned by user `student`, and whose modification time is greater than 120 minutes ago, and have find prompt you interactively whether or not to remove each one. Because you are using the interactive option, do not throw out error messages; that is, do not end your command with 2> /dev/null Decline to remove all files when prompted.

find /tmp -user student -mmin +120 -type f -ok rm {} \;

10 *CHALLENGE PROBLEM:*

Use `find` to locate each regular file in `/home` that is owned by `student` and pass the matching filenames to **tar**, adding them to a bzipped archive

HINT: Do not try to solve this problem with **-exec**. You want to pass the filenames all at once instead of one at a time, so use **xargs** instead. The **xargs** command is discussed near the end of Unit 5. Using tar is discussed in Unit 8.

find /home -type f -user student | xargs tar -jcvf student.tar.bz2

Copyright © 2006 Red Hat, Inc.
All rights reserved

Sequence 1 Solutions

1. **find /var/lib -user rpm 2> /dev/null**
2. **find /var -user root -group mail 2> /dev/null**
3. **find / -not -user root -not -user bin -not -user student -ls 2> /dev/null**

   or

   find / ! -user root ! -user bin ! -user student -ls 2>/dev/null
4. **find /usr/bin -size +50k -ls 2> /dev/null**
5. **find /etc/mail -exec file {} \; 2> /dev/null**
6. **find /etc/ -type l -ls 2> /dev/null**
7. **find /tmp -user student -mmin +120 -type f -ls 2> /dev/null**
8. **find /bin /usr/bin -perm -4000 2> /dev/null**

   or

   find /bin /usr/bin -perm -u+s 2> /dev/null
9. **find /tmp -user student -mmin +120 -type f -ok rm {} \;**

   Note: The standard error is not redirected in answer number 8 because that would prevent the questions being asked by **-ok** from being displayed.
10. **find /home -user student | xargs tar -jcvf mystuff.tar.bz2**

Copyright © 2006 Red Hat, Inc
All rights reserved

Unit 15

Investigating and Managing Processes

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Objectives

Upon completion of this unit, you should:

- Understand what a process is
- Understand process states
- Know how to manage processes
- Understand job control

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved.

What is a Process?

- A process has many components and properties
  - exec thread
  - PID
  - priority
  - memory context
  - environment
  - file descriptors
  - security credentials of user who is running it.

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Processes

A process is an executing program with several components and properties.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

How Processes Are Created

- One process "forks" a child, pointing to the same pages of memory, and marking the area as read-only
- Then, the child "execs" the new command, causing a copy-on-write fault, thus copying to a new area of memory
- A process can exec, without forking
  - The child maintains the process ID of the parent

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Creation of Processes

Traditionally, in Unix-like operating systems, processes began through a process called "fork and exec" That is, when one command starts another, the child process first forks, the kernel copying over pages of memory from the parent process to a new location for the child process. The child then execs, executing the new command, overwriting the data. For example, when the shell executes the ls command, the kernel would duplicate the shell's pages of memory and then execute the ls command, overwriting the duplicated data. This was wasteful of system resources.

Linux replaced this model for process initialization with three other methods, the most commonly used being copy-on-write. With copy-on-write, when one process starts a subprocess, the original process does not duplicate its pages of memory when it forks, but rather it points the child to the same pages in memory as the original process, but marking the memory area as read-only for both parent and child. When either process wishes to write to the data, as will happen when the child execs, a copy-on-write fault will occur and the kernel will create new pages in memory for that process.

Manually exec'ing Processes

It is also possible for a command to exec without forking For example, when logging into the X Window System, if the `.xsession` file exists, the `.xsession` file will execute the commands in a shell. The last command will be something like:

```
exec metacity
```

This means the shell in which the `.xsession` command was running will be replaced in memory with the **metacity** command. Were the command to fork and exec, the shell that ran `.xsession` would remain on the system performing no useful function until the **metacity** command exited.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Process Ancestry

- **init** is the first process started at boot time - always has PID 1
- Except **init**, every process has a parent
- Processes can be both a parent and a child at the same time

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Viewing Process Ancestry

pstree shows the process ancestry for all process running on the system

```
[student@stationX ~]$ pstree
init-+-apmd
|-atd
|-2*[automount]
|-battery_applet
|-bdflush
|-bonobo-moniker-
|-cardmgr
|-crond
|-deskguide_apple
|-dhclient
|-galeon-bin---galeon-bin---4*[galeon-bin]
|-gconfd-1
|-gdm---gdm-+-X
|            `-gnome-session
|-gnome-name-serv
|-gnome-smproxy
|-gnome-terminal-+-bash---tail
|                |-bash
|                |-bash---less
|                |-bash---ssh
|                |-bash---pstree
|                |-bash---vim
|                `-gnome-pty-helpe
|-gpm
|-kapmd
|-keventd
...output truncated...
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Process States

- A process can be in one of many states
  - R - Runnable (on the run queue)
  - S - Sleeping
  - T - Stopped
  - D - Uninterruptible sleep
  - Z - Defunct (zombie) process

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Process States

The following process states are defined in Linux:

- *Runnable*: Process is on the run queue It is waiting for its turn to run or it is executing.
- *Sleeping*: Process is not executing, nor is it ready to run It is waiting for an event to occur or a signal to arrive to wake it up.
- *Stopped*: Process is not executing because it has been stopped.
- *Uninterruptible sleep*: Process is sleeping and can not be woken up until an event occurs. It can not be woken up by a signal. Typically, the result of an I/O operation.
- *Zombie*: Just before a process dies, it sends a signal to its parent and waits for an acknowledgment before terminating. Even if the parent process does not immediately acknowledge this signal, all resources except for the process identity number (PID) are released Zombie processes are cleared from the system during the next system reboot and do not adversely affect system performance

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Viewing Processes

- **ps**
  - Displays processes information
  - Syntax: **ps [options]**
  - Useful options:
    - **a** Processes by all users
    - **x** Processes from all terminals
    - **u** Show process owners
    - **w** Include command arguments
    - **f** Show process ancestry

-Ls show remote process.

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Viewing All Processes

The **ps** command provides several styles of output depending on the options used. Without options, it displays information about processes specific to the active terminal. The options described below are based on output conforming to the UNIX98 standard. Consult the online documentation for more options

-a	displays all processes, not including processes not controlled by a terminal
-x	includes processes not controlled by a terminal, such as daemon processes.

Formatting output

-l	displays a "long" listing, which includes more information such as the process owner's uid
-u	displays the user name of the process owner

Viewing Specific Process Information

Since there may be hundreds of processes on a system, a common technique to locate a specific process is to send output from **ps** to **grep**:

```
[student@stationX ~]$ ps -alx | grep 'lpd'
```

The above command will display all the lines of the **ps** command's output that contain lpd. If the only line of output is the **grep** process itself, chances are that lpd is not running. Alternately, the **pgrep** command:

```
[student@stationX ~]$ pgrep lpd
```

will list only the PIDs of matching processes.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sending Signals to Processes

- Syntax:
  - **kill [signal] pid(s)**
  - **kill [signal] %jobID**
  - Sends the specified signal to a process
  - Default signal is TERM
  - **kill -l** lists all available signals
- **killall**

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Sending signals

Signals can be specified by their name, such as KILL, or by their number, such as 9. (A detailed list of signals can be displayed with **man 7 signal**.)

kill can send many signals, but processes only respond to the signals they have been programmed to recognize.

The following are all identical and will send the default TERM signal to the process with PID number 3428:

```
[student@stationX ~]$ kill 3428
[student@stationX ~]$ kill -15 3428
[student@stationX ~]$ kill -TERM 3428
```

These commands:

```
[student@stationX ~]$ kill -9 3428
[student@stationX ~]$ kill -KILL 3428
```

are likewise identical and will send a KILL signal to the process

In addition to **kill**, there is a **killall** command that can be used to send a signal to a group of commands, such as all **getty** processes. It should be used with caution since killing some processes may have a detrimental effect on the system. Here's an example that sends a KILL signal to all running processes named **galeon-bin**:

```
[student@stationX ~]$ killall -KILL galeon-bin
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

Signals may be sent to processes interactively using **top** or gnome-system-monitor (Applications->System Tools->System Monitor.)

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Terminating Processes

- Most desirable way to end a process is to let it end normally
  - Commands finish, applications are exited
- Can attempt to interrupt with *Ctrl-c* or send a TERM signal
- If all else fails, send a KILL signal

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Terminating Processes

Processes ordinarily terminate on their own when they have completed their task. Interactive applications may need the user to issue a `quit` command.

Many processes can be terminated with *Ctrl-c*, which sends an interrupt (INT) signal to the process. The process is shutdown "cleanly": that is, child processes are terminated first and any pending I/O operations are completed The same is true if a process is sent a terminate (TERM) signal via the **kill** command.

If a process will not respond to a TERM signal, the KILL signal can be used. However, the process may not be ended cleanly. The KILL signal should be used only if a process will not respond to a *Ctrl-c* or a TERM signal. Using KILL signals on a routine basis may cause zombie processes and lost data.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Altering Process Scheduling Priority

- At process invocation time
  - Syntax:
    - **nice [-n *priority*] command**
- Processes are scheduled with a default nice value of 0
- Nice values can range from -20 (highest priority) to 19 (lowest)

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Schedule priorities

Every running process has a scheduling priority: a ranking among running processes determining which should get the attention of the processor. The formula for calculating this priority is complex, but users can affect the priority by setting the "niceness" value, one element of this complex formula The niceness value is a number ranging from -20 (highest priority) to 19 (lowest priority). It defaults to a value of zero

Specifying priorities for programs

The **nice** command is used to modify this default niceness value Example:

```
[student@stationX ~]$ nice myprog
```

This runs the **myprog** program with a niceness value of 10. To set the niceness value to a different value, use the **-n** option:

```
[student@stationX ~]$ nice -n 15 myprog
```

Non-privileged users may not set niceness value to less than zero: that is, they may not request a higher than normal priority for their processes. This is a function reserved for the superuser

Copyright © 2006 Red Hat, Inc
All rights reserved

Altering Process Scheduling Priority (continued)

- **renice** changes the priority of a running process
  - **renice *priority* [[-p|-g] PID] [[-u]user]**
- Once a priority value is raised, a non-privileged user cannot lower it

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2594 or +1 (919) 754 3700.

Altering priorities of running programs

Users may reduce the priority of currently running jobs using the renice command:

```
[student@stationX ~]$ renice 15 -p PID
```

Only the superuser is permitted to raise the priority of currently running processes:

```
[root@stationX ~]# renice -15 -p PID
```

The **-p** option is not strictly necessary. However, if you wish to change the priority of an entire process group, use the **-g** option. Also useful is the **-u** option, which can be used to modify the priority of all of the processes of a particular user:

```
[student@stationX ~]$
          renice 15 -u student
```

Of course, this only works if you actually are user student; non-privileged users may not alter the priority of other users' processes.

Interactive Process Management Tools

- Display real-time process information
- Allow sorting, killing and renicing
- Command-line: **top**
- GUI: gnome-system-monitor

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*The **top** command*

Running **top** presents a list of the processes running on your system, updated every 5 seconds. You can use keystrokes to kill, renice and change the sorting order of processes. Processes that are in the Running state are highlighted and you can colorize processes and create multiple windows to view more than one sorted list of processes at a time. Press the *?* key while in top to view the complete list of hotkeys. You can exit top by pressing the *q* key.

The gnome-system-monitor command

The gnome-system-monitor, which can be run from the console or by selecting ApplicationsSystem ToolsSystem Monitor from the menu system, is a graphical tool that also offers the ability to view sorted lists of processes and to kill or renice those processes.

By default the tool only displays processes that are in the "Running" state. However, this can be changed by selecting All Processes or My Processes from the dropdown menu in the upper-right. The preferences dialog (EditPreferences) allows you to customize which fields are displayed and you can sort by a particular field by clicking on its heading. You can send a process the TERM signal by right-clicking on it and selecting End Process or the KILL signal (equivalent to a **kill -9**) by selecting Kill Process A process can be re-niced by right-clicking on it and selecting Change Priority

Copyright © 2006 Red Hat, Inc.
All rights reserved

Running a Process in the Foreground

- When a command is entered, the shell will not process further input until the process is complete and the shell prompt is redisplayed
- The typeahead buffer allows you to type other commands, but they will not be processed until the pending process completes, or "returns"

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Multi-tasking at the command line

When a process is started from a terminal's command line, it is normally running in the foreground. The process can be stopped, restarted in the background, or terminated This is called "job control."

Job control is often necessary on text-based systems where it is impossible to open another terminal window Under XOrg, it is often just as easy to open another terminal.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Running a Process in the Background

- Running a command in the background allows another process to run concurrently on the same terminal
- Launch a program as a background process by appending an ampersand (&) to the end of a command:
  - **mozilla &**

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Background processes

A background process is still the child of the processes that spawned it. The parent process, however, does not wait for the child process to terminate before continuing

Background processes that generate standard output and standard error often have these redirected.

When a process is started in the background, a new bash "subshell" is created The **bash** program is then replaced with the command being executed (the fork then exec procedure) Background processes can be managed like any other process

Suspending a Process

- Foreground jobs can be suspended: temporarily halted without being killed
- Suspend a foreground process with *Ctrl-z*
- Suspended jobs can be:
  - Resumed in the background (**bg**)
  - Resumed in the foreground **fg**)

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Suspending a Job

Jobs running in the foreground can be suspended: temporarily stopped, without being killed. To suspend a job, use the keys *Ctrl-z*. Once a process is suspended, it can be resumed in the background, using the **bg**bg command, or resumed in the foreground, using the **fg** command. When the job resumes, it will continue executing from the point at which it was suspended; it will not have to start over from the beginning. Example:

```
[student@stationX ~]$ find / -name '*.conf' -print >/tmp/conf.list 2>&1
[student@stationX ~]$ Ctrl-z
[1]+  Stopped                 find / -name '*.conf' -print >/tmp/conf.list 2>
[student@stationX ~]$ bg
[1]+ find / -name '*.conf' -print >/tmp/conf.list 2>&1 &
```

Typically, a job will be resumed in the background because the user intended to start the job in the background in the first place, but forgot to provide the ampersand background symbol. Alternately, the user may not have realized how long the job would take to run

Also, typically, a job will be suspended and then resumed in the foreground because the user needs to temporarily jump out of the job for some reason. Perhaps the user is editing a file using **vi** and then suspends **vi** while looking up some other information on the system. The user would then resume **vi** using the **fg** command.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Listing Background and Suspended Jobs

- **jobs** displays all process running in the background or that are suspended
- The number in brackets is a job number, used to kill jobs or bring them back to the foreground
- Job numbers are referenced with %

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Viewing your jobs

jobs also reports the status of suspended and backgrounded processes. The job ID reported can be used with other commands to manage the process.

```
[student@stationX ~]$ jobs
[1]+  Stopped                 man bash
[2]-  Running                 find / -name joe >output 2>&1 &
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Resuming Suspended Jobs

- When a command is suspended or backgrounded, it can be brought back to the foreground with **fg**
- Suspended jobs can be resumed in the background with **bg**
- Syntax:
  - **fg [%job_number]**
  - **bg [%job_number]**

17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Resuming jobs in the background

Using a *Ctrl-z* and **bg** combination is useful when a time-consuming process (or one that opens its own window) is started in the foreground

For example, the following sequence will place the foreground find process in the background:

```
[student@stationX ~]$ find / -name '*.ps' 2> /dev/null > ps.out
[student@stationX ~]$ Ctrl-z
[student@stationX ~]$ bg %1
```

`%1` above refers to job ID number one in the shell. The job ID number may vary if there are other jobs running in the same shell. Use jobs to verify the job ID number, if in doubt

Compound Commands

- List of commands separated by semi-colons
- List inside () to run inside a sub-shell
  - $ (cd /usr; du) &

18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Compound Commands

Suppose you want to maintain a count of the number of users logged on, along with a time/date stamp, in a log file. This could be accomplished with two commands:

```
[student@stationX ~]$ date >> logfile
[student@stationX ~]$ who | wc -l >> logfile
```

This command sequence requires that you enter two lines of commands, append to logfile twice, and in general, type much more than is necessary. When writing to the terminal, this task can be simplified by combining the commands on one line separated by semicolons:

```
[student@stationX ~]$ date; who | wc -l
```

But if your intent is to redirect standard input, this will not work as expected:

```
[student@stationX ~]$ date; who | wc -l >> logfile
```

Both commands will run, but only the second one will redirect its output to logfile.

A sub-shell group will combine the commands so they are treated them as one unit. When a group of commands is placed inside parentheses, a new sub-shell is spawned and output can be redirected as if it were one command.

```
[student@stationX ~]$ (date; who | wc -l) >> logfile
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Scheduling a Process To Execute Later

- Syntax:
  - **at** *time*
    - *commands*
  - **atq** *user*
  - **atrm** *user*|*atJobID*
- Commands will be executed at the time indicated
  - Non-redirected output is mailed to the user

19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

*Using the **at** command*

Commands are entered, one per line, and terminated with a *Ctrl-d* on a line by itself

The `time` argument has many formats which are illustrated by the following examples. See the online documentation for more information

- **at 8:00pm December 7**
- **at 7 am Thursday**
- **at now + 5 minutes**
- **at midnight + 23 minutes**

atq lists the current at jobs pending. A privileged user can supply a user ID argument to obtain the pending at jobs of other users.

atrm is used to remove pending at jobs. The atJobID is displayed when the job is submitted and also from **atq**. A privileged user can remove the at jobs of other users by supplying the user ID.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Scheduling Periodic Processes

- The **cron** mechanism allows processes to be invoked periodically
  - User need not be logged on
- Cron jobs are listed in a `crontab` file

20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*Using **cron** to schedule processes*

The **cron** mechanism is controlled by a process named **crond**. This process runs every minute and determines if an entry in users' cron tables need to be executed. If the time has passed for a entry to be started, it is started. A cron job can be scheduled as often as once a minute or as infrequently as once a year

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Using cron

- Must edit and install your **cron** file
- The **cron** file cannot be edited directly
  - Edit the file and then install with **crontab**
- Syntax:
  - **crontab [-u user]**
  - **crontab [-l|-r|-e]**

21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*Using **cron** to schedule processes (continued)*

Cron files ("crontabs") are stored in `/var/spool/cron,` which is not accessible by non-privileged users. In order to access the current crontab, the **crontab** command is used.

The crontab can either be edited in the current directory and installed by using it as an argument to crontab or by using **-e**

The **-l** option displays the current crontab file and the **-r** option removes it.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Crontab File Format

- Entry consists of five space-delimited fields followed by a command line
  - One entry per line, no limit to line length
- Fields are minute, hour, day of month, month, and day of week
- Comment lines begin with #

22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Crontab examples

Entry fields can be separated by any number of tabs or spaces. Valid field values are as follows:

Minute 0-59

Hour 0-23

Day of Month 1-31

Month 1-12

Day of Week 0-6 (0 = Sunday)

Multiple values may be separated by commas. An asterisk in a field represents all valid values A user's crontab may look like the following:

```
#Min   Hour   DoM   Month   DoW     Command
0      4      *     *       1,3,5   find ~ -name core | xargs rm -f {}
0      0      31    10      *       mail -s "boo" $LOGNAME < boo.txt
```

End of Unit 15

- Questions and Answers
- Summary
  - Understand what a process is
  - Understand process states
  - Manage processes
  - Understand job control

23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc
All rights reserved

Lab 15
Process Control

Goal: Practice using the various process control related commands.

Estimated Duration: 30 minutes

System Setup: A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Sequence 1: Process Control

Scenario: In this sequence you will start several processes running, then use the job control functions of bash to manage and control them. You will be switching between virtual consoles. Please make careful note of which console is used for each command.

Instructions:

1. Start by logging in on virtual terminals 1 and 2 (`tty1`, `tty2`) using the `student` account.

2. Switch to `tty1` and start a process running with the following command:

```
[student@stationX ~]$ (while true; do echo -n A >> log; ↙
sleep 1; done)
```

3. Notice that this terminal is now busy with your process (running in the foreground.) The process that you have started is appending the letter "A" repeatedly at one second intervals to the file `~/log`. To provide visual evidence of this, switch to `tty2` and execute the following:

```
[student@stationX ~]$ tail -f log
```

   You should see a sequence of "A" characters that is increasing in length.

4. Switch to `tty1` and suspend the running process by pressing *Ctrl-z* at that console. The shell reports that the process has been stopped, and gives you a job number of [1] for the job. Switch to `tty2` and visually verify that the file is no longer growing.

5. Switch to `tty1` and start the process running again, this time in the background. Use jobs to show that job [1] reports as running. Finally, switch to `tty2` and visually verify that the file is again growing:

```
[student@stationX ~]$ bg
[student@stationX ~]$ jobs
....output omitted....
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

6. Switch to tty1 and start two more processes running with the following commands. (The first command is almost identical to the command you ran in step 2. Use the *Up Arrow* to recall that command, change the "A" to a "B", and append the ampersand to the end The second command simply runs the previous command, substituting a "C" for the "B"):

```
[student@stationX ~]$ (while true; do echo -n B >> log; ↵
sleep 1; done) &
[student@stationX ~]$ ^B^C
```

7. Type jobs and verify that all three are running Switch to tty2 and visually verify that the file is again growing, this time with "A", "B", and "C" all being added every second.

8. In step 4, you suspended the foreground process by pressing *Ctrl - z*. In reality, this sent a signal to the process. Use kill to produce a list of signals and their corresponding names and numbers. Then use kill to send a SIGSTOP to job [1] to suspend it. Switch to tty1 and execute the following:

```
[student@stationX ~]$ kill -l
[student@stationX ~]$ kill -19 %1
```

9. Type **jobs** and verify that job [1] reports stopped. Switch to tty2 and visually verify that job [1] has stopped.

10. Restart job [1] by using **kill** to send the process a SIGCONT (18) Use the jobs command and tty2 to verify that all three jobs are again running

    Hint: refer to step 8 for help on the syntax

11. kill all three processes. If you do not specify a signal to send, kill will send a SIGTERM (15) by default. Catching a SIGTERM will cause most programs to terminate. After signaling jobs [2] and [3], use jobs to verify that they report terminated:

```
[student@stationX ~]$ kill %2 %3
[student@stationX ~]$ jobs
```

12. To terminate the final process:

```
[student@stationX ~]$ fg
[student@stationX ~]$ Ctrl-c
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

13. Type **jobs** and verify that no jobs are listed. Switch to `tty2` and visually verify that activity has stopped. Stop the tail command by pressing *Ctrl-c*, and log out of this tty (**exit**).

14. Switch to `tty1` and clean up by deleting the `~/log` file

Copyright © 2006 Red Hat, Inc
All rights reserved

Unit 16

bash Shell Scripting

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Be comfortable automating tasks with shell scripting

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc
All rights reserved

Scripting Basics

- Shell scripts are text files that contain a series of commands or statements to be executed.
- Shell scripts are useful for:
  - Automating commonly used commands
  - Performing system administration and troubleshooting
  - Creating simple applications
  - Manipulation of text or files

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

A shell script is simply a text file containing commands. Scripts are useful for automating processes that you perform repeatedly at the command line. For example, suppose every morning when you log in, you perform the following operations:

- Check the system date.
- Look at the calendar for the current month.
- Check your email.
- Display the list of logged in users.

Instead of entering the commands to perform steps 1-4 individually, you could create a shell script containing those command. Every morning you could enter just one command to perform these steps. In addition, one of the key benefits to using Linux is its powerful, yet simple commands. These simple commands may be put together in a script to perform complex operations or automate procedures such as adding batches of users

Linux users involved with system administration and troubleshooting often work with shell scripts. Many of these are created during the installation of the operating system. Consider the script `/etc/profile`. This file is the system-wide user login script which runs whenever a user logs into the system. System administrations and troubleshooters will consider operations performed in this script when troubleshooting user login problems.

Programmers often create simple versions of programs using scripts during the initial phases of a programming project. This is called *application prototyping*. Once they and the program's users are happy with the main functionality of the program, the programmer will then create the full-featured program in a programming language such as C.

Creating Shell Scripts

- Step 1: Use a text editor such as **vi** to create a text file containing commands
  - First line contains the magic "shbang" sequence: #!
    - `#!/bin/bash`
- Comment your scripts!
  - Comments start with a #

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Shell scripts are text files that generally contain one command per line, but you can have multiple commands on a line if you separate them with semicolons (;) In order to continue a command onto the next line you use the line continuation character. For the Bourne shell(/bin/sh) and its derived shells such as bash, this is a backslash followed by a newline You can enter this by pressing the \ key followed by the ***Enter*** key on most keyboards This will enable you to enter one command that spans multiple lines.

The first line in a shell scripts should contain 'magic', which is commonly referred to as the *shbang*. This tells the operating system which interpreter to use in order to execute the script. Some examples are:

Shbang Use

- **`#!/bin/bash`** - used for Bash scripts (most common on Linux)
- **`#!/bin/sh`** - used for Bourne shell scripts (common on all UNIX-like systems)
- **`#!/bin/csh`** - used for C shell scripts (common on BSD derived systems)
- **`#!/usr/bin/perl`** - used for Perl scripts (an advanced scripting and programing language)
- **`#!/usr/bin/python`** - used for Python scripts (an object oriented programming language)

Commenting Shell Scripts

It is extremely important to put comments in your shell scripts Anything following the # symbol is a comment and therefore ignored by the interpreter. It is good practice to make a shell script self documenting Self documenting means that someone with almost no scripting knowledge can read your comments in the script and reasonably understand what the script does. The easiest way to do this is to write the comments before you actually write the code. This has an additional benefit of clarifying to yourself, what your objective is. In addition, this practice makes the script easier to maintain for both yourself and others As time progresses, you may not recall what you were trying to do at the time and comments may help clarify the original objective.

Creating Shell Scripts continued

- Step 2: Make the script executable:

```
$ chmod u+x myscript.sh
```

- To execute the new script:
  - Place the script file in a directory in the executable path -OR-
  - Specify the absolute or relative path to the script on the command line

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

After you have created the script and saved it using a text editor, you will need to change its file permissions in order to make it executable (Note that scripts must also be readable to run so that the shell can read the script and interpret it into executable code.)

An example of making a script you own executable:

```
[student@stationX ~]$ chmod u+x scriptfile
```

-or-

```
chmod 750 scriptfile
```

Ensure that the script is located in a directory listed by the PATH environmental variable. To do this, enter the following command:

```
[student@stationX ~]$ echo $PATH
```

If the script is not in a directory listed in the PATH variable, either move the script to a directory that is (such as $HOME/bin) or specify the absolute or relative path on the command line when executing the script:

```
[student@stationX ~]$ /home/user/mytestscript
```

or

```
[student@stationX ~]$ ./mytestscript
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

Generating Output

- Use **echo** to generate simple output
  - `echo 'Welcome to Red Hat Linux paradise!'`
  - `echo -n "Please enter the file name: "`
- Use **printf** to generate formatted output
  - Syntax similar to C **printf()**function
  - Does not automatically put a newline at the end of the output.
    - `printf "The result is %0.2f\n" $RESULT`

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

printf

printf provides additional formatting flexibility for output compared with the echo command Unlike echo, printf does not provide a newline This allows multiple printf statements to apply to one line for complex formatting situations The escape sequence `\n` must be used if a newline is desired.

The format string used by printf can consist of literal characters, format control strings and additional options. Format control strings are in the form *%width.precision* followed by a conversion character. The value *width* specifies the minimum field width The value *precision* sets the number of digits to display after the decimal for numbers displayed in scientific notation or floating point numbers. For character strings, *precision* sets the maximum number of characters that will be displayed Some of the possible conversion characters are: (**c** - character, **d** - integer, **f** - single or double precision floating point number , **e** - exponential notation, **s** - string).

Example:

```
#!/bin/sh
var1="Testing"
var2=12345
var3=6.789
printf "var1 is %10.5s\n" $var1
printf "var1 is %7.7s and " $var1 # this and next output on same line
printf "var2 is %5.5e\n" $var2
printf "var3 is $%05.2f\n" $var3
```

Output:

```
var1 is      Testi
```

```
var1 is Testing and var2 is 1.23450e+04
var3 is $06.79
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

Handling Input

- Use **read** to assign input values to one or more shell variables:
  - **-p** designates prompt to display
  - **read** reads from standard input and assigns one word to each variable
  - Any leftover words are assigned to the last variable
  - `read -p "Enter a filename: " FILE`

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

read takes a line from standard input and breaks it down into individual words. (Usually a word is defined as a character string surrounded by white space such as spaces and tabs). The way the shell interprets words may be changed by setting the `IFS` variable (e.g. **`IFS=':'`** will tell the shell that words are separated by colons instead of white space). The first word is assigned to the first variable and the second word is assigned to second variable, and so on. If there are more words than variables, the last variable is assigned all the remaining words

Example:

```
#!/bin/bash
read -p "Enter name ( first last ): " FIRST LAST
echo "Your first name is $FIRST and your last name is $LAST"
```

The (-p) option is used to display a prompt string. Place quotes around the string if you need to prompt the user with a multiple-word command

Example:

```
#!/bin/bash
read -p "Enter several values:" value1 value2
value3
echo "value1 is $value1"
echo "value2 is $value2"
echo "value3 is $value3"
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Exit Status

- Commands exit with an exit status
  - 0 for success, 1 to 255 for failure
  - Exit status of most recently executed command is kept in the **$?** variable just like return values from shell functions
- Shell scripts may set an exit status with the **exit** command:
  - **exit 1 # Indicates an error**

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Upon completion, every command returns an exit status. The exit status will be a number in the range of 0 to 255 and it indicates whether or not the command ran successfully. As a general rule, an exit status of 0 indicates success and an exit value in the range of 1 to 255 indicates failure. To see the exit status of the most recently executed command, echo the **$?** variable:

```
[student@stationX ~]$ ls -d /tmp
/tmp
[student@stationX ~]$ echo $?
0
[student@stationX ~]$ ls -d /tpm
ls: /tpm: No such file or directory
[student@stationX ~]$ echo $?
1
```

In the example above, the first **ls** command exited successfully, but the second **ls** command had a typo in the argument and so failed. The exit statuses of each command demonstrates this.

Command exit statuses are important because they are used to determine actions when conditionally executing commands using **if** statements or when running commands in a loop using **while** or **until** loops, as we shall see in coming pages

It is also possible for a shell script to deliberately set the exit value. Usually, a shell script will exit with the exit status of the last command run in the script. But an exit status can be force by giving a numeric argument to the **exit** command. For example, this command will exit with an exit status of 255:

```
exit 255
```

Copyright © 2006 Red Hat, Inc
All rights reserved

Conditional Execution
if Statements

- Executes instructions based on the exit status of a command

```
if ping -c1 server1
then
   echo "The server is UP"
else
   echo "The server is DOWN"
fi
```

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

So-called "selection structures" allow arbitrary code to be executed based on a logical decision. This is also known as conditional execution. For example, if something is true, do one thing and if it is false do something else. The most basis such structure in bash is the **if** statement:

```
read -p "Enter a username: " USER
if grep "^$USER" /etc/passwd &>/dev/null
then
   echo "$USER exists"
fi
```

Remember that every command on a Red Hat Enterprise Linux system returns a true or false (zero or non-zero) value, so any command can be used as a condition in an **if** statement. In the example above we use the **grep** command to determine whether a given username exists in `/etc/passwd`. Note that all output is redirected to `/dev/null` because in this case we don't care about the command's output, only its return value. If grep returns 0, which will only happen if it succeeds in finding the user, we will print a message. Otherwise nothing will happen.

But what if you want to print one message if the command succeeded and another if it failed? That is where **else** comes in. Our example could be altered to incorporate an **else** statement as follows:

```
read -p "Enter a username: " USER
if grep "^$USER" /etc/passwd &>/dev/null
then
   echo "$USER exists"
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

```
else
   echo "$USER does not exist"
fi
```

Think of this example as reading: "**if** the command succeeds, **then** do one thing Otherwise (**else**) do another"

Copyright © 2006 Red Hat, Inc
All rights reserved.

Conditional Execution shorthand

- Commands can also be run conditionally without **if** statements
  - **&&** represents logical AND
  - **||** represents logical OR
- Examples:

```
$ grep nonexistant_user passwd || echo 'No such user!'
No such user!

$ cp -a /tmp/*.o . && echo 'Done!' || echo 'Failed!'
Done!
```

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

It is possible to run commands conditionally; depending on the exit status of another command. When executing two commands separated by **&&**, the latter command will only run if the former command exits successfully (exits with a status of zero). When executing two commands separated by **||**, the second command runs if the first command failed (it exits with an exit status in the range of 1 to 255) For example:

```
$ grep joe passwd || echo 'No joe!'
```

This command will search the **passwd** file for the string "**joe**". The **grep** command sets the exit status to '0' if it finds any match and to '1', indicating an error, if it finds no match. Therefore, '**No joe!**' will be echoed to the screen if the string "**joe**" is not found in the **passwd** file

The **&&** and **||** are sometimes called "logical AND" and "logical OR", respectively. You can think of conditional syntax in this way:

thiscmd && thatcmd run thiscmd successfully AND **then run thatcmd**

firstcmd* || *secondcmd run **firstcmd** successfully OR run **secondcmd**

Conditional execution is often done in shell scripts using this syntax, or using other syntax shown on the pages that follow. In addition it is common to use this structure on the command line as well.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Conditional Execution
the test command

- Describes true/false statements for use in conditional execution
- Options determine the type of condition
- Returns 0 or 1 based on outcome of the condition
- Can be specified using the **test** command:

```
$ test "$A" = "$B" && echo "Values are equal"
```

- Or using shorthand:

```
$ [ "$A" = "$B" ] && echo "Values are equal"
```

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

The **test** command can be be used to describe true-or-false scenarios for use in conjunction with conditional statements. **test** takes a number of arguments which allow it to test everything from numerical comparisons to whether a file is executable or not Several of these arguments will be discussed in the proceeding slides.

A complete list of arguments can be obtained from the **test** man page

Copyright © 2006 Red Hat, Inc
All rights reserved

Conditional Execution
file tests

- File tests:
  - **-f** tests to see if a file exists and is a regular file
  - **-d** tests to see if a file exists and is a directory
  - **-x** tests to see if a file exists and is executable

```
if [ -f ~/lib/functions ];then
        source ~/lib/functions
fi
```

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

File tests can be used to test a variety of conditions that relate to files on the system. This provides an easy mechanism to test for the existence of files, conditions or permissions. Some of the supported file tests are:

-d *FILE*	True if the file is a directory
-e *FILE*	True if the file exists.
-f *FILE*	True if the file exists and is a regular file.
-h *FILE*	True if the file is a symbolic link.
-L *FILE*	True if the file is a symbolic link
-r *FILE*	True if the file is readable by you
-s *FILE*	True if the file exists and is not empty.
-w *FILE*	True if the file is writable by you.
-x *FILE*	True if the file is executable by you.
-O *FILE*	True if the file is effectively owned by you.
-G *FILE*	True if the file is effectively owned by your group.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Conditional Execution
string tests

- Strings may be tested as well
  - **-z** returns true if the string is empty
  - **-n** returns true if the string is not empty
  - operators such as **=**, **!=**, **<** and **>** may be used to compare strings as well

```
if [ $(id -u) = "0" ]; then
        echo "You are logged in as root"
fi
```

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Strings may also be tested to make sure they are not empty or compared against other strings. The string tests available are:

-z ***STRING***	True if *STRING* is empty.
-n ***STRING***	True if *STRING* is not empty
STRING1 = ***STRING2***	True if the *STRING*s are equal.
STRING1 != ***STRING2***	True if the *STRING*s are not equal.
STRING1 < ***STRING2***	True if *STRING1* sorts before *STRING2* lexicographically.
STRING1 > ***STRING2***	True if *STRING1* sorts after *STRING2* lexicographically.

Other tests are available and numeric equations may be tested as well:

-o ***OPTION***	True if the shell option *OPTION* is enabled
! ***EXPR***	True if *EXPR* is false.
EXPR1 -a ***EXPR2***	True if both *EXPR1* AND *EXPR2* are true.
EXPR1 -o ***EXPR2***	True if either *EXPR1* OR *EXPR2* is true.
ARG1 OP ARG2	Arithmetic tests. *OP* is one of -eq, -ne, -lt, -le, -gt, or -ge.

Copyright © 2006 Red Hat, Inc
All rights reserved.

for loops

- Performs actions on each member of a set of values
- Example:

```
for NAME in joe jane julie
do
   ADDRESS="$NAME@example.com"
   MESSAGE='Projects are due today!'
   echo $MESSAGE | mail -s Reminder $ADDRESS
done
```

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

for loops are useful for performing the same action on each member of a set of values. The basic syntax of a **for** loop is:

```
for variable in value1 value2...
do
   command using $variable
   ...
done
```

This would perform the specified command(s) once for each item in the value list. The first time through `variable` would be equal to `value1`, the second time to `value2` and so forth. Any number of values and any number of commands can be specified.

Value lists can be space-delimited sets of values, as shown above, the output of a command:

```
for USER in $(grep bash /etc/passwd)...
```

or a file glob:

```
for FILE in *.txt...
```

for loops
numeric sequences

- **for** loops are useful for iterating through numeric sequences
  - Use bash notation for simple sequences: **for i in {0..10}**
    - Will use: 0,1,2,3,4,5,6,7,8,9,10
  - Use **seq** command for arbitrary increments: **for i in $(seq 0 2 10)**
    - Will count by twos: 0,2,4,6,8,10

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

It is often useful to either perform a loop a specific number of times or to loop through a sequence of numbers. Such tasks can be accomplished easily using for loops and either bash's builtin sequence notation or the **seq** command. Bash's sequence notation is simple enough: **{0..100}** is equivalent to listing the numbers 0 through 100. The seq command is a little more powerful in that it allows you to optionally specify an increment other than one For example: **seq 0 10** 100 would count by tens from 0 to 100. The example below is the same as the example on the previous slide but uses a sequence to test all machines from 192 168.0 1 to 192.168.0.20

Example:

```
#!/bin/bash
# alive2.sh
# Checks to see if hosts 192.168.0.1-192.168.0.20 are alive
# Iterate through IP addresses
for n in {1..20}; do
        host=192.168.0.$n
        ping -c2 $host &> /dev/null
        if [ $? = 0 ]; then
                printf "%-30s is alive\n" $host
        else
                printf "%-30s is NOT alive\n" $host
        fi
done
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

while loops

- **while** loops perform commands until a condition has been met
- Example:

```
while [ "$INPUT" != "n" ]
do
    . . commands . .
   read -p "Do you want to go again? [Y/n]: " INPUT
done
```

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Sometimes, especially when taking instructions from users, such as in a menu-driven application, it is helpful to execute instructions until a particular condition is met, such as the user entering an exit command The basic syntax of a **while** loop looks like this:

```
while condition
do
   commands
   ...
done
```

Copyright © 2006 Red Hat, Inc
All rights reserved.

while loops
continue and break

- **while** loops can be disrupted during execution
  - **continue** stops the current execution of the loop and reexamines the initial condition, possibly restarting the loop
  - **break** stops processing the loop entirely, jumping past the done statement
  - **exit** exits from the shell script entirely
    - You may provide an exit status

17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Sometimes, a condition can be detected within a while or until loop that indicates that the loop should stop executing. Three commands can be used to disrupt the execution of a loop: **continue**, **break**, and **exit**.

The **continue** command stops the current execution of the loop, jumps back to the initial condition (before the **do** command), and reruns the condition to see if another run through the loop is needed.

The **break** command stops the execution of the loop entirely and jumps to the command past the done command. This is needed if, for example, you have an infinite loop such as:

```
echo -n "This will run forever "
while true
do
        echo -n "and ever "
done
```

In this example, the **true** command will never fail; to exit the loop it will be necessary to use a **break** (or **exit**) command. Without a **break** or **exit**, the code above would print `This will run forever and ever and ever and ever and ever...` until manually stopped with a *Ctrl-c*.

Occasionally, in the middle of a loop (or elsewhere in the shell script), it will be necessary to exit. This can be accomplished with the **exit** command. If the exit is abnormal in some way, you can provide an exit status by following the exit command with a non-zero number. Without an explicit exit status, the **exit** command will exit with at status of zero, indicating success

Copyright © 2006 Red Hat, Inc.
All rights reserved

Positional Parameters

- Positional parameters are special variables that hold the command-line arguments to the script.
- The positional parameters available are `$1`, `$2`, `$3`, etc.. These are normally assigned to more meaningful variable names to improve clarity.
- `$*` holds all command-line arguments

18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Many commands and scripts can perform different tasks depending on the arguments supplied to the program. These values are assigned to positional parameters that may be accessed in the script. The variable `$0` is reserved and specifies the program name as it was executed on the command line. Variables above `$9` require special handling and so they must be enclosed in curly braces, e.g. `${11}`.

All positional parameters are read only variables. In most cases they are reassigned to other more meaningful names which improve the readability of the script. In addition, reassigning the variables allows read - write access to the information contained within the positional parameter.

Example:

```
#!/bin/bash
# positionaltester
# Demonstrates the use of positional parameters
###########################################################################
echo "The program name is $0"
printf "The first argument is %s and the second is %s\n" $1 $2
echo -e "\nAll command line parameters are $*\n"
```

Output:

```
[student@stationX ~]$ ./positionaltester Red Hat Linux
The program name is ./positionaltester
The first argument is Red and the second is Hat
All command line parameters are Red Hat Linux
```

Copyright © 2006 Red Hat, Inc
All rights reserved.

Positional Parameters
handling spaces

- Bash expects space-delimited parameters
  - Causes problems when parameters have spaces
  - Example: $ **script.sh "arg 1" "arg 2"**
    - `$*` contains "arg" "1" "arg" "2"
- Solution: **for VAR; do ...; done**
  - Automatically assigns `VAR` with `$1`, `$2`, etc...
    - Handles spaces in parameters gracefully
    - `VAR` would be set to "arg 1", then "arg 2"

19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Suppose you wanted to write a script that parsed command-line arguments. Something like the function below would work fine unless the arguments had spaces Note the output below the example.

```
#!/bin/sh
# say_hello.sh
for P in $*
do
        echo "Hello $P"
done
```

Output

```
[student@stationX ~]$ say_hello.sh "fred flinstone" "barney rubble"
Hello fred
Hello flinstone
Hello barney
Hello rubble
```

A better solution is to use bash's for loop, which will automatically loop through parameters correctly:

```
#!/bin/sh
# say_hello2.sh
for P
do
        echo "Hello $P"
```

Copyright © 2006 Red Hat, Inc
All rights reserved

```
done
```

Output:

```
[student@stationX ~]$ say_hello2.sh "fred flinstone" "barney rubble"
Hello fred flinstone
Hello barney rubble
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Scripting at the Command Line

- Scripts can be typed at the bash prompt
  - **if/case/for/ while** statements cause extended prompt
  - Can also separate lines with semicolon
- bash builtin **fc** for long commands
  - Default editor is **vi** or `$EDITOR`

20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The simplest way to script on the command line is just to type it out like you would in a program. You can either string your commands out on one line, separated by semicolons:

```
[student@localhost ~]$ if [ "$PWD" = "$HOME"]; then echo 'Welcome home!';fi
Welcome home!
```

Or you can put it on multiple lines and let bash handle it:

```
[student@localhost ~]$ if [ "$PWD" = "$HOME"]
> then
> echo 'Welcome home!'
> fi
Welcome home!
```

If you're doing the same stuff over and over, you should consider defining a few functions These can be placed in your bashrc to automatically load them when you log in:

As you script more on the command line you may find that commands become too long to reasonably edit on the command line alone. To help with this bash provides the **fc** builtin By itself it will open up the previous command in an editor If given a numeric argument, it will open that command from your history

Copyright © 2006 Red Hat, Inc
All rights reserved.

Shell script debugging

- In order to debug a shell script invoke the shell interpreter with debug options or change the shebang to include the debug options
  - **bash -x** *scriptname*
  - **bash -v** *scriptname*
  - **#!/bin/bash -x**
  - **#!/bin/bash -v**

21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

The bash shell may be invoked to provide additional output which may prove useful when debugging shell scripts. Either you may use bash directly on the command line.

Example:

```
[student@stationX ~]$ bash -x email.sh emails textmessage
+ emailaddressfile=emails
+ message=textmessage
+ read emailaddress
+ cat emails
+ cat textmessage
+ mail -s 'Important email' foo
+ read emailaddress
+ cat textmessage
+ mail -s 'Important email' bar
+ read emailaddress
```

Or, the shbang may be changed to include the debugging options eg:

```
#!/bin/bash -v
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 16

- Questions and Answers
- Summary
  - The **bash** shell includes a powerful scripting language
  - Scripts can be written in executable files or on the command line
  - Custom functions can be stored in `~/.bashrc` and run like commands

22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved

Lab 16
bash Shell Scripting

Goal:	Become proficient with the basics of Bash shell scripting
Estimated Duration:	60 minutes
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Sequence 1: Reading bash scripts

Instructions:

1. Read the following script:

```
#!/bin/sh
MYNAME=$(/bin/hostname)
read -p "What is your name?" YOURNAME
echo "Hello, $YOURNAME. My name is $MYNAME"
```

2. Describe what this script does If you're having trouble, you can always copy these lines into a file, make it executable (chmod u+x filename), and test the script on your workstation.

   Hello, Jeff. My name is Localhost, Station 7

3. How might you rewrite the script's last line so that it prints out the date along with its current output?

   echo "Hello, $YOURNAME. My name is $MYNAME The date is $(date)"

4. Read the following script:

```
#!/bin/sh
MIN=1
MAX=20
for n in $(seq $MIN $MAX)
do
        ping -c1 -w2 192.168.0.$n &> /dev/null
        if [ $? -eq 0 ]
        then
                echo "Station$n is UP"
        fi
done
```

5. Describe what this script does. How might it be useful?

   HINT: If you aren't familiar with the arguments given to the **ping** command, check the man page!

   HINT: Check your courseware if you don't remember what '$?' means. It's very important!

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 2: Writing Simple Scripts

Deliverable: Two simple shell scripts: one that prints your name and another that translates strings to upper-case

Instructions:

1 Write a bash script that prints out your name. It can be as simple or as complicated as you want, as long as your name ends up being printed to the screen.

2 Suppose you find yourself regularly using the tr command to convert strings to upper case as in the following example:

```
[student@stationX ~]$ echo "one" | tr a-z A-Z
ONE
```

To save yourself some typing and make the command a bit more intuitive, write a **bash** script called **up.sh** that uses **tr** to convert all of its arguments to upper case. The result should behave something like this:

```
[student@stationX ~]$ up.sh "one"
ONE
```

3. Now try modifying up.sh so that it can take multiple parameters. The script's output when passed multiple values should look something like this:

```
[student@stationX ~]$ up.sh "one" "two" "three"
ONE
TWO
THREE
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

4. To see how well your script handles more complicated parameters, try giving it an argument with spaces:

```
[student@stationX ~]$ up.sh "hello there"
```

It should return

```
HELLO THERE
```

Instead of

```
HELLO
THERE
```

If it does not, consider how you could improve the parameter processing of your script and try to fix it.

Copyright © 2006 Red Hat, Inc
All rights reserved

Challenge Sequence 3:

Instructions:

1. If there is time remaining, write something cool and impress your instructor!

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 1 Solutions

2. This script first runs the command **/bin/hostname**, which simply prints they system's hostname, and stores the command's output in a variable called MYNAME. It then prompts the user for his or her name using the prompt command and stores the response in a variable called YOURNAME. Finally, it prints out the value of each variable. So if the user Jane ran this command on a system called station1 then the output would be:

```
Hello Jane. My name is station1
```

3. You could modify the last line of this script to print the date by interpolating the output of the date command into the final echo command. For example:

```
echo "Hello $YOURNAME. My name is $MYNAME. The current date ✐
and time is $(date)."
```

5. This script pings a range of IP addresses and returns a list of which ones are up and which ones are down. First, two variables called `MIN` and `MAX` are set to define the range of addresses to **ping**. Next, a **for** loop is used to cycle through all the numbers between `MIN` and `MAX` (the **seq** command is used to generate this list of numbers). During each iteration of the loop, the ip address ending in the current number is pinged. Then the return value of the **ping** command is tested by looking at the `$?` variable (remember that `$?` always contains the return value of the most recent command). If the return value is zero, indicating success, then a message is printed reporting that the system is up. If the **ping** fails, no action is taken. Then the loop repeats until every number between `MIN` and `MAX` has been checked.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Sequence 2 Solutions

1. The solution to this problem could be a simple as

```
#!/bin/sh
echo "Linus Torvalds"
```

A more complex (and therefore cooler) solution might be something like the following, which can get the name to be printed from an environment variable, a command-line argument, prompting the user or a default value:

```
#!/bin/sh
# If the name was set as an environment variable we'll use tha
# but if not then we'll have to look elsewhere
if [ -z "$NAME" ]; then
        # Does it look like the name was provided as an argume
        if [ -n "$1" ]; then
                NAME="$1"
        # If all else fails, ask the user
        else
                read -p "What is your name? " NAME
                # If the user just hits Enter, use a default
                if [ -z "$NAME" ]; then
                        NAME=?Joe User?
                fi
    fi
fi
echo "Hello $NAME"
```

2 Since this script is essentially a simpler way to echo a string to `tr`, it can be very short:

```
#!/bin/sh
# Remember that $1 represents the first command-line argument
echo $1 | tr a-z A-Z
```

3 To handle multiple command-line arguments we will need to modify the script to run **tr** on each argument. A `for` loop can handle this task nicely but we must be careful how we loop through the arguments. A solution like the following would suffice, but will not handle spaces nicely. A solution that handles spaces will be demonstrated in answer number 4.

```
#!/bin/bash
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

```
for ARG in $*
do
        echo $ARG | tr 'a-z' 'A-Z'
done
```

4 Remember that when looping through command-line arguments using `$*`, **bash** assumes that the arguments are separated by spaces. This causes problems when an argument includes spaces in its value. The following solution would do the same thing as the previous solution with the added bonus of handling spaces gracefully (as long as the value that contains spaces is given in quotes).

```
#!/bin/bash
for ARG
do
        echo $ARG | tr 'a-z' 'A-Z'
done
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Unit 17

Network Clients

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Objectives

Upon completion of this unit, you should:

- Know how to browse the web
- Be able to exchange email and instant messages
- Know how to access a Linux system remotely
- Know how to transfer files between systems
- Be familiar with the use of network diagnostic tools

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc
All rights reserved.

Web Clients

- Firefox
- Other web browsers
- Non-GUI web browsers
- **wget**

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Firefox

- Fast, lightweight, feature-rich web browser
  - Tabbed browsing
  - Popup blocking
  - Cookie management
  - Multi-engine search bar
  - Support for many popular plugins
  - Themes and Extensions

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Firefox Features

Firefox supports one of the newer things in web browsing: Tabbed browsing. Tabbed browsing allows a user to quickly navigate and access multiple web pages in a single browser window. When you create a bookmark by selecting Bookmarks->Bookmark This Page you will notice a checkbox called Bookmark All Tabs in a Folder. If this is selected then all open pages will be bookmarked as a group. Selecting the group from the Bookmarks menu will allow you to re-open the tabs at a later date.

Firefox also automatically blocks annoying popup windows When this happens, Firefox will display a message letting you know what has happened and giving you the option to hide the message, view the popup and/or allow popups in the future from this site The Edit->Preferences->Web Settings dialog allows you to customize which websites Firefox allows popups from

With the advent of additional toolbars and the Google toolbar for IE, Firefox has included a multifunctional search toolbar that allows for searching multiple engines and the addition of user defined engines. A list of available engine-plugins you can add is at `http://mycroft.mozdev.org`

In the past, plugins for browsers on linux have required the separate download and running of files to install for most browsers. Firefox includes the hot-plugin capability for most of its plugins so that the user does not have to download and hand install them (flash, etc) A list of available plugins can be found at `http://plugindoc.mozdev.org`

In many cases a user might wish to customize the look and feel of their browser. This functionality is done through 'theming'. You can download and manage themes with the the Tools->Themes dialog.

In addition to changing the way Firefox looks with themes, you can add whole new elements of functionality to the browser by adding extensions. Available extensions include enhancements for bookmark managing, web searching, html validation, chatting, blogging and more. To download and manage extensions, select the Tools->Extensions option

Copyright © 2006 Red Hat, Inc
All rights reserved.

Other GUI Web Browsers

- Epiphany
  - Uses same rendering engine as Firefox
  - Fully Gnome-compliant, but fewer features
- Konqueror
  - KDE web browser/file manager
  - Supports tabs, popup-blocking, etc
  - Uses khtml rendering engine

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc.
All rights reserved

Non-GUI Web Browsers

- **links**
  - Provided by the *elinks* rpm
  - Full support for frames and ssl
  - Examples
    - **links http://www.redhat.com**
    - **links -dump http://www.redhat.com**
    - **links -source http://www.redhat.com**

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Browsing the web without a GUI: ***links***

A number of graphical web clients are provided, discussed in earlier slides, but Red Hat also provides a text-based web client called **links** This tool can be used on the command line to test network availability as well as provide a fast and stable utility for browsing web pages without graphics. It is capable of retrieving files via http and ftp sites, and can be used to download a web page non-interactively. It is frames-capable and has built in support for SSL

Useful Arguments to links

The **-dump** argument causes links to print the text of a page to the standard output and then exit This provides a fast and convenient way to test web connectivity or retrieve text-based files from websites. Another useful argument is **-source**, which does the same thing as **-dump**, but with the web page's html source rather than the rendered content

Copyright © 2006 Red Hat, Inc.
All rights reserved.

wget

- Retrieves files via HTTP and FTP
- Non-interactive - useful in shell scripts
- Can follow links and traverse directory trees on the remote server - useful for mirroring web and FTP sites

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

*Automated http and ftp retrieval: **wget***

wget can be used to retrieve a single file via HTTP or FTP:

```
[student@stationX ~]$ wget http://www.redhat.com/training/index.html
--17:17:59--  http://www.redhat.com/training/index.html
=> `index.html'
Resolving www.redhat.com... done.
Connecting to www.redhat.com[66.187.232.56]:80... connected.
HTTP request sent, awaiting response... 200 OK
...output truncated...
17:18:00 (295.44 KB/s) - `index.html' saved [28438]
```

wget's "retry" options can be useful when fetching files from busy ftp sites:

```
[student@stationX ~]$ wget --tries=50 --wait=30 ftp://ftp.site.com/files
```

When using **wget** to mirror a full or partial web site, you can limit number of levels of recursion (link following):

```
[student@stationX ~]$ wget --recursive --level=1 --convert-links http://www.s
```

See **wget --help** and **info wget** for many more options.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Email and Messaging

- Evolution
- Other email clients
- Non-GUI email clients
- Gaim

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Evolution

- Default email and groupware tool
- Provides email, calendar, tasks and contacts
- Can maintain multiple accounts at once
- Supports GnuPG encryption and signatures
- "Trainable" bayesian spam filters
- Task/Calendar notifications in Gnome clock
- Can sync with many PDAs

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Email Protocols

Mail clients require mail servers Typically different server types will be used to send mail from those used to deliver mail. Mail pickup is typically done through a member of either the IMAP, POP, or Microsoft Exchange family of protocols. The most popular variants of IMAP/POP are IMAPS and POP3S, forms of the protocol that encrypt data over the wire. Mail delivery is most typically done through one of the SMTP family of protocols, either SMTP or ESMTP. Microsoft Exchange also supports its own variant of these. Email clients must be configured to use the proper protocols for pickup and delivery.

Evolution Email and Groupware Features

The default tool for email and groupware in Red Hat Enterprise Linux is called Evolution Evolution provides not only email access, but the ability to maintain a calendar, tasklist and contacts database Your can share your calendar and access shared calendars through a variety of methods: LDAP directories, Novell Groupware, Microsoft Exchange or just publishing your calendar to a website When you click on the clock icon in Gnome's panel (in the upper-right of the screen) a summary of all of the day's tasks and calendar appointments will appear In addition to all this, Evolution can sync with many different PDA operating systems including PalmOS.

Security and Anti-Spam Features

Evolution also supports encrypted email using the Gnu Privacy Guard (GnuPG) and has powerful filters for managing spam. Bayesian spam filters help cut down on junk mail by allowing the user to make unwanted email as junk. This allows the filters to become increasingly personalized and accurate over time.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Configuring Evolution

- Defining accounts
  - Tools->Settings->Mail Accounts
  - Supports IMAP, pop, Novell Groupware, Usenet and local email accounts
  - MS Exchange support via plugin
    - Provided by evolution-connector rpm
    - Install *before* configuring other accounts

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Interoperability with Exchange Server

The Evolution email client can be configured to communicate with Microsoft Exchange servers via the Evolution-connector plugin. This must be separately installed as it does not default install. One current issue with the Evolution exchange connector is that it will only allow you to configure your email with an Exchange server on the initial setup for your account. If you wish to add an Exchange server later on you must export your current mail and re-import it after you have cleared out the exchange folder in your home directory The user must know the name of the Exchange server that he/she is connecting to in order to configure this option as there is no dropdown list for servers

Email and Encryption

- Problem: Email is normally sent unencrypted
- Solution: Asymmetric Encryption
  - Two "keys" are generated
    - Public key: Used to encrypt messages to you
    - Private key: Used to decrypt messages from public key
  - Keep private key, distribute public key
- Exchange public keys with others for two-way encryption

11

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Asymmetric Encryption

Most classic encryption algorithms use a secret value, called a key, to scramble messages. Running the message and key through one algorithm returns an encrypted version of the message Running the encrypted message (also called the *cyphertext*) and the key through another algorithm returns the original message This allows all parties who have access to the key to encrypt and decrypt messages for each other These single-key algorithms are called *symmetric* algorithms.

The problem with symmetric encryption is that in order to use it one must communicate the secret key to all involved parties. But how does one do this in a secure manner? It can't be encrypted-- what would you encrypt it with? But if the key is communicated unencrypted then anyone who intercepts it will be able to eavesdrop on your *secure* communications

Asymmetric or *public/private key* algorithms avoid this problem by using two keys instead of one One key, called the *public key* is used for encrypting messages to you These messages can only be decrypted with another key called the *private key*. Using an asymmetric algorithm you keep your private key secret but give your public key to anyone who wants it. Anyone who has your public key can then encrypt messages that only you can decrypt. If you exchange public keys with another then you can encrypt messages with her public key and she can encrypt replies with your public key.

Tools that use or can be used for asymmetric encryption in Red Hat Enterprise Linux include **gpg, openssl** and **ssh**.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Email and Digital Signatures

- Problem: Forging the source of an email is trivial
- Solution: Digital signatures
  - A unique *hash* of your message is generated
  - Hash is scrambled using private key and attached
  - Recipient descrambles hash with your public key
    - If this fails, the signature must have been forged
  - Recipient recalculates message hash and checks for a match
    - If this fails, message must have been altered

12

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Digital Signatures

It is an unfortunate fact that forging the source address of an email is extremely easy to do. The best way to defend against this is to consistently include some verifiable proof of your authorship of an email other than the source address. Digital signatures provide this ability. Digitally signing an email is a three-step process:

First, a *hash* of the email is generated. A hash is a one-way scrambling of the contents of the message. It is different from encryption in that a hash cannot be reversed. The significance of a hash is that every time you send the message through a particular hashing algorithm you should always get back the same result. This will be used later to determine if the message has been altered en route.

Once a hash has been generated, it is scrambled using your private key (note: not your public key as would normally be the case). This scrambling is such that it can only be reversed with your public key. Thus, if someone receives a message that claims to be signed by you, but is unable to descramble the message hash with your public key then the recipient can conclude that the message is actually from someone else pretending to be you.

Finally, the scrambled hash is attached to the email and the email is sent. Upon receipt, the recipient's email client (assuming it supports digital signatures) will look at who the message claims to be signed by and check to see if it knows that person's public key. If so then the email client will descramble the hash using the appropriate key. If this is successful then the email client knows that it was sent by the person it claims to have been sent by (or by someone with access to the sender's private key). Next, the email client will generate a hash of the message using the same algorithm that created the signature hash. If the hashes match then the client knows that the text of the message received is the same as when the message was signed. The recipient will then be informed whether or not the signature check was successful.

Copyright © 2006 Red Hat, Inc
All rights reserved

Evolution and GnuPG

- GnuPG is a powerful encryption tool
- Evolution can use GnuPG keys to encrypt and/or digitally sign email
- To generate a key: **gpg --gen-key**
- To associate your key with an account:Tools->Settings->Accounts->Edit->Security
- Exchange public keys with others to encrypt email and verify signatures

13

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Using gpg

The Gnu Privacy Guard (GnuPG) allows you to generate keypairs for use in asymmetric cryptography. In other words, GnuPG and Evolution can be used together to send digitally signed and/or encrypted email. First, you will need to create a keypair This is done with the command: **gpg --gen-key**. You will be prompted for information about yourself, which will be associated with the key. This information will include the email address to be associated with the keys and a passphrase that will be required in order to sign, encrypt or decrypt anything using them. While the passphrase may be left blank, it is highly recommended that you set one. Otherwise, someone who compromised your account would also be able to view encrypted documents and send signed emails as you

Once configured to do so Evolution can use GnuPG to encrypt, decrypt and sign emails transparently, but the gpg command can be used to perform these operations manually on any file. Some examples:

gpg -a -o me@example.com.pubkey --export exports your public key to a file

gpg --import joe@example.com.pubkey imports joe's key from a file into your collection

gpg -a -r joe@example.com --sign --encrypt ***filename*** encrypts and signs a file for joe

gpg -a -o me@example.com.pubkey --export exports your public key to a file

gpg --import joe@example.com.pubkey imports joe's key from a file into your collection

gpg --decrypt ***filename*** decrypts and verifies the signature on a file

Configuring Evolution to use GnuPG

Once you have created a key you will need to let Evolution know about it. Select the Tools->Settings menu option, then from the **Accounts** section select your primary account, click **Edit** and then go to the

Security tab. In the field marked **PGP/GPG Key ID**, enter the email address associated with the key that you wish to use and click **Ok**. Your emails will now be signed automatically and you can encrypt mail to anyone whose public key you have by selecting **Security/PGP Encrypt** while composing an email How to share your key and add others' keys to your key collection (called your *keyring*) will be discussed during the lab.

Copyright © 2006 Red Hat, Inc
All rights reserved.

Other GUI Mail Clients

- Thunderbird
  - Standalone Mozilla email client
- Kmail
  - KDE email client

14

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat Inc
All rights reserved.

Non-GUI Mail Clients

- **mutt**
  - Supports pop, imap and local mailboxes
  - Highly configurable
  - Mappable hotkeys
  - Message threading and colorizing
  - GnuPG integration
  - Context-sensitive help with '?'

15

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Introducing Mutt

It is important to know how to access email even when a graphical interface is not available. The **mutt** email client is a powerful tool for managing email within a text-only environment. Many Linux users even prefer it over evolution. Some of the strengths of **mutt** are its handling of message threads and its flexibility. Every hotkey can be mapped to your liking and whole strings of actions can be bound to a single keystroke When **mutt** sees a series of related messages (an original message, followed by a series of replies on a mailing list, for example), it displays each reply under the parent. These *threads* can then be navigated, deleted and marked as read or unread as a group instead of one at a time. This makes managing high volumes of email much easier.

Mailboxes in mutt

You read one "mailbox" (a local mail spool, imap account or pop account) at a time when using **mutt** You can specify the mailbox you wish to start in by running mutt with the **-f** argument. For example:

```
[student@stationX ~]$ mutt -f imaps://user@server
```

If no mailbox is specified then your local mail spool will be viewed. While in mutt you can switch mailboxes by pressing the **c** key and typing the url of the mailbox you would like to read. You can change which mailbox mutt views by default by altering its configuration file, `~/.muttrc`

Documentation and Help

Documentation and example `.muttrc` files are available in mutt's `/usr/share/doc/` subdirectory mutt has a large number of very powerful keybindings for managing large amounts of email. Fortunately it also has context-sensitive help to assist you in finding out how to perform a given task. For example, if you press the **?** key while viewing the message-list for a mailbox, the keybindings for managing a mailbox will be displayed. If you press **?** while viewing a message then the keybindings for navigating a message will be displayed. The most commonly used commands will always be displayed at the top of your screen.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Gaim

- Multi-protocol Instant messaging client
- Supports AIM, MSN, ICQ, Yahoo, Jabber, Gadu-Gadu, SILC, GroupWise Messenger, IRC and Zephyr networks.
- Plugins can be used to add functionality.

16

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Instant messaging and more: Gaim

Gaim users can log in to accounts on multiple IM networks simultaneously. This means that you can be chatting with friends on AOL Instant Messenger, talking to a friend on Yahoo Messenger, and sitting in an IRC channel all at the same time. Gaim supports file transfers, away messages, typing notification, and buddy pounces Plugins can be used to add a buddy ticker, extended message notification, iconify on away, spell checking, tabbed conversations, integration with Evolution and more. Many of these plugins come with Gaim and can be configured in the Tools -> Preferences -> Plugins dialog. More plugins, implementing features like end-to-end encryption, are available at `http://gaim.sourceforge.net/plugins.php`

Copyright © 2006 Red Hat, Inc
All rights reserved.

ssh: Secure Shell

- Secure replacement for older remote-access tools
- Allows authenticated, encrypted access to remote systems
  - **ssh [*user*@]*hostname***
  - **ssh [*user*@]*hostname command***

17

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Using the secure shell

ssh allows remote logins and remote command execution via a secure encrypted connection. For example, to execute **df -h** on the host **hork**:

```
[student@stationX ~]$ ssh hork 'df -h'
jmh@hork's password:
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda9             252M  137M  102M  57% /
/dev/hda1              19M  5.5M   12M  31% /boot
/dev/hda6             508M   29M  453M   6% /tmp
/dev/hda5             2.0G  1.7G  221M  88% /usr
/dev/hda7             252M  119M  120M  50% /var
/dev/hdc1             2.3G  1.9G  298M  87% /home
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

scp: Secure copy

- Secure replacement for rcp
- Layered on top of ssh
  - **scp** ***source destination***
  - Remote files can be specified using:
    - **[user@]host:/path/to/file**
  - Use **-r** to enable recursion
  - Use **-p** to preserve times and permissions
  - Use **-C** to compress datastream

18

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

scp

scp works like **cp**, except that it copies from one host to another in a secure encrypted channel.

```
[student@stationX ~]$ scp hork:proclog .
jmh@hork's password:
proclog              100% |*****************************|
28117     00:00 ETA
```

Also available is **sftp**, an interactive file-transfer program similar to a simple ftp client. The remote host's sshd needs to have support for **sftp** in order for the this to work

scp requires that the destination be a directory if the source is a directory or consists of more than one file.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

telnet and the "r" services

- Insecure protocols mostly replaced by ssh
  - **telnet**: Login names and passwords pass over the network in clear text
  - "**r**" services (**rsh, rlogin, rcp**): generally insecure authentication mechanism
- **telnet** can be used to connect to arbitrary ports for testing
  - Example: testing your mail server:
  - **telnet localhost 25**

19

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Legacy protocols and tools

The telnet protocol is inherently insecure: login names and passwords pass over the network in clear text, making them readily detectable by anyone with physical access to the network The "r" tools have the same problem, plus further issues with the underlying authentication mechanism. These tools should never be used.

Telnet as a diagnostic tool

The telnet client, however, has other functions telnet is useful for checking and troubleshooting services. For example, you can telnet port 25 to see if sendmail is running on your local machine:

```
[student@stationX ~]$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost
Escape character is '^]'.
220 localhost.localdomain ESMTP Sendmail 8.11.6/8.11.4;
Sat, 22 Sep 2001 16:10:11 -0400
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

rsync

- A drop-in replacement for **rcp** for copying to or from remote systems
- Can use **ssh** for transport
  - **rsync -e ssh *.conf barney:/home/joe/configs/**
- Faster than **scp** - copies differences in like files

20

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Efficient network copies: rsync

rsync is a program that works in much the same way that **rcp** does, but has many more options and uses the **rsync** remote-update protocol to greatly increase the speed of file transfers when the destination file already exists.

The **rsync** remote-update protocol allows **rsync** to transfer just the differences between two sets of files using an efficient checksum-search algorithm. This utility is useful for tasks like updating web content, because it will only transfer the changed files.

Useful options to **rsync**

-e *command*	specifies an external, rsh-compatible program to connect with (usually **ssh**)
-a	recurses subdirectories, preserving permissions, ownership, etc
-r	recurses subdirectories without preserving permissions, etc.
--partial	continues partially downloaded files
--progress	prints a progress bar while transferring
-P	is the same as **--partial --progress**

See the rsync(1) man page for a complete list

Copyright © 2006 Red Hat, Inc.
All rights reserved

lftp

- Versatile command-line FTP client
- Anonymous or real-user sessions
  - **lftp ftp.cdrom.com**
  - **lftp -u joe ftp.myserver.com**
- Automated transfers with **lftpget**

21

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Command-line ftp: ***lftp***

Several FTP clients are available, including the traditional, if somewhat fundamental, **ftp**. More featureful clients include **lftp**, the non-interactive **lftpget**, and the graphical **gftp**.

The **lftp** client includes such useful features as bookmarks and tab completion. Below is an example of using lftpget non-interactively:

```
[student@stationX ~]$ lftpget ftp://ftp.example.com/pub/file.txt
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

gFTP

- Graphical FTP client
- Allows Drag-and-Drop transfers
- Anonymous or authenticated access
- Optional secure transfer via ssh (sftp)
- Applications->Internet->gFTP

22

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Graphical FTP: gFTP

gFTP is a graphical FTP client that can be used to upload and download files from remote servers. It can handle authentication using anonymous or real user access. If the SSH2 protocol is chosen, the authentication process as well as all transmitted data is encrypted. If you have public key authentication configured, gFTP can use it rather than a password when authenticating.

Copyright © 2006 Red Hat, Inc.
All rights reserved

smbclient

- FTP-like client to access SMB/CIFS resources
- Examples:
  - **smbclient -L server1** lists shares on server1
  - **smbclient -U student //server1/homes** accesses a share

23

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Accessing Network-Neighborhood style shares: smbclient

smbclient is a command-line tool that provides access to SMB/CIFS (most commonly implemented as Microsoft Windows Network Neighborhood) shares

Useful options include:

-W	workgroup or domain
-U	username
-N	suppress password prompt (otherwise you will be asked for a password)

Once connected, **smbclient** behaves very much like an ftp client: You can navigate using **cd** and **ls**, upload and download using **put** and **get**, etc

Copyright © 2006 Red Hat, Inc.
All rights reserved.

File Transfer with Nautilus

- File/Connect to Server
- Graphically browse with multiple protocols
- Allows drag-and-drop file transfers
- Supported connection types: FTP, SFTP, SMB, WebDAV, Secure WebDAV
- Can also connect via url:
  - File/Open Location

24

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Nautilus for accessing network services

Nautilus not only navigates local filesystems, but can be used to connect to remote file shares. Selecting "Connect to Server" from the File menu will open a dialog where you can enter details about the connection you would like to make, including the protocol, server and username that you would like to use.

Nautilus URL examples

Once you get used to using the different protocols, you may find that typing in a url is faster and simpler than filling out a form. In browser mode accessing sites by url is as simple as typing in the Location bar. In spatial mode you can enter a url by selecting "Open Location" from the File menu. Some example urls:

- `ftp://ftp.example.com` connects via ftp to ftp.example.com
- `sftp://user@ftp.example.com` is the same as above, but connecting securely via ssh as 'user'
- `smb://` lists all available SMB servers
- `smb://example` lists shares on the example server
- `smb://example/share` accesses a particular share on the server
- `smb://user@example/share` is the same as above, but authenticates as "user"

Copyright © 2006 Red Hat, Inc
All rights reserved.

Xorg Clients

- All graphical applications are X clients
  - Can connect to remote X severs via tcp/ip
  - Data is not encrypted but can be tunneled securely over an **ssh** connection
    - **ssh -X hostB[user@hostB]$ xterm &**
- **xterm** will display on hostA's X server
- Transmitted data will be encrypted through the **ssh** connection

25

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Network transparency in Xorg

One of the most important features of Xorg is that its client/server architecture makes any graphical application on Red Hat Enterprise Linux completely network-transparent. After connecting to a remote host, configuring access controls and setting some environment variables, any graphical application (also called an X client because it communicates with the Xorg server using the X protocol) that is run will display on the local screen. Applications running in this way will still be running on the remote system They will take up resources (cpu cycles, memory, etc) on the remote system and affect files on the remote system. The application is only being displayed on the local system.

Running remote X clients with ssh

One problem with doing this over older remote-access methods, such as **telnet**, is that X clients do not encrypt the data that they send to the Xorg server. So, for example, if you were to run a remote **xterm** and then use the **su** command to become `root`, when you typed the password it would be sent across the network in plain text. This problem can be circumvented by using **ssh** as your remote-access tool because it can be configured to automatically set up a "tunnel" that forwards all X communications over the established, encrypted ssh session Another advantage of using **ssh** for running remote X clients is the fact that all necessary access controls and environment variables will configured for you automatically

However, the **ssh** client does not do this automatically Unlike in in previous versions of Red Hat Enterprise Linux, the ssh client must be told explicitly that you wish to enable *X forwarding* with the **-X** option If you do not plan to run X applications on the remote system, it is a best practice to not enable X forwarding as doing so allows a higher level of access to your system from the (possibly hostile) remote system than a normal ssh session would.

Copyright © 2006 Red Hat, Inc
All rights reserved

Network Diagnostic Tools

- **ping**
- **traceroute**
- **host**
- **dig**
- **netstat**
- **gnome-nettool** (GUI)

26

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Basic network diagnostic tools

A number of network diagnostic tools are available:

ping

detects if it is possible to communicate with another system Many systems no longer respond to pings

traceroute

displays the computers through which a packet must pass to reach another system. The mtr command is a repetitive version of traceroute, giving continually updated connection time statistics

host

performs hostname to IP address translations, as well as the reverse

dig

performs a service similar to host in greater detail.

netstat

provides a number of network statistics.

gnome-nettool

a graphical frontend to the tools listed above (as well as some others) in a single, simple interface. gnome-nettool can be run from the command line or by selecting its icon from the Internet section of the Applications Menu. Note that this tool may not be installed by default.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

End of Unit 17

- Questions and Answers
- Summary
  - Firefox, Evolution and Mutt
  - Basic network diagnostic tools
  - The importance of secure network clients

27

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Lab 17
Network Clients

Goal:	Practice using a variety of tools to transfer files between your system and a remote system
Estimated Duration:	1.5 Hours
System Setup:	A working, installed Red Hat Enterprise Linux system with an unprivileged user account named `student` with a password of *`student`*.
Lab Setup:	Assign a student to act as the data repository. Install and enable an ftp server on the student system. Also, start the dovecot imap service on server1

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 1: Transferring files with lftp

Scenario: One system will act as a repository for data from other computers. The remaining computers will store their data on the data repository and the synchronize their own data with that of the data repository.

Deliverable: A modified / renamed text file transferred to the ~student/data directory on the repository system

Lab Setup: Instructor should enable the sendmail daemon on server1, make sure dovecot is running, set up one of the student machines as an ftp server and make sure there is a directory /home/student/data

Instructions:

1. One student's system will act as the data repository. The student whose system will perform this task should ensure that the student account on this system has a password of student and a directory called **data**. For the remainder of this lab, this system will be called stationY

2. The remaining students should verify network connectivity with stationY, using the ping command:

```
[student@stationX ~]$ ping -c 3 stationY
PING stationY.example.com (192.168.0.Y) from 192.168.0.2 ....
64 bytes from stationY.example.com (192.168.0.Y): icmp_seq=0 ...
64 bytes from stationY.example.com (192.168.0.Y): icmp_seq=1 ...
64 bytes from stationY.example.com (192.168.0.Y): icmp_seq=2 ...

--- stationY.example.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.212/0.214/0.218/0.017 ms
```

3. Use **lftp** to connect anonymously to server1 and get a file:

```
[student@stationX ~]$ cd
[student@stationX ~]$ lftp server1
lftp server1:~> cd pub
lftp server1:/pub> ls
-rw-r--r--    1 0        0            26 Jun 13 23:57 getme
lftp server1:/pub> get getme
26 bytes transferred in 2 seconds (13b/s)
lftp server1:/pub> exit
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

4. Examine, and then modify the text file that you have retrieved:

```
[student@stationX ~]$ cat getme
{Your name here} was here!
[student@stationX ~]$ vi getme
```

Insert your name where indicated, then save the file as `getme.XY` where XY=your initials.

```
[student@stationX ~]$ cat getme.bd
Bob Dobalina was here!
```

5. Use **lftp** to connect to the repository system, stationX, as user `student`, with a password of *`student`*, and transfer your modified file into that user's `~/data` directory:

```
[student@stationX ~]$ lftp -u student stationX
Password:     type_password_here

lftp server1:~> cd data
lftp student@stationX:/> put getme.bd
21 bytes transferred.

lftp student@stationX:/> exit
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 2: Encrypted communication -- The ssh suite

Scenario: In this sequence, you will use the **ssh** suite of utilities to securely transfer a file between your machine and the data repository You will then establish an encrypted login session with the remote host, and verify that your file was successfully transferred.

Instructions:

1. Start by making a copy of your `getme.XY` file used in the last sequence (Be sure to replace the sample filename listed below with your initials):

```
[student@stationX ~]$ cd
[student@stationX ~]$ cp getme.bd getme.bd.secure
```

2. Securely transfer your new file via an encrypted session back to the data directory of user student on the repository system:

```
[student@stationX ~]$ scp getme.bd.secure student@stationY:data
... output omitted ...
```

3. Establish an encrypted session to the repository system, and verify that your file has been successfully transferred:

```
[student@stationX ~]$ ssh student@stationY
student@stationY.example.com's password: student
[student@stationY student]$ ls data/*bd*
getme.bd  getme.bd.secure
[student@stationY student]$ exit
```

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Sequence 3: Synchronizing your files with a remote system

Scenario: In this sequence, you will use the **rsync** command to perform a sync with several files on the repository system.

Deliverable: Synchronization between the student `data` directory on the repository system and your local home directory.

Instructions:

1. Perform a sync operation against the home directory on the repository system, transferring new and changed files to your local system:

```
[student@stationX ~]$ cd
[student@stationX ~]$ rsync -e ssh student@stationY:data/get* .
student@stationY's password: student
[student@stationX ~]$ ls getme*
getme.af          getme.ai          getme.bg          getme.cf
getme.af.secure   getme.ai.secure   getme.bg.secure   getme.cf.sec
getme.ag          getme.bc          getme.bh          getme.cg
getme.ag.secure   getme.bc.secure   getme.bh.secure   getme.cg.sec
...output truncated...
```

Copyright © 2006 Red Hat, Inc
All rights reserved.

Sequence 4: Configuring evolution

Instructions:

1. While logged into the X Window System, change to your home directory and list your file:

```
[student@stationX ~]$ cd
[student@stationX ~]$ ls -aF
```

2. Start evolution from the Red Hat menu: Applications->Internet->Evolution Email

3. The first time evolution starts, it will run you through a setup wizard. Select the **Forward** button

4. The next screen is the "Identity" screen. Input your full name in the proper place.

5. The next field is where your "Email address" gets entered. For this class, your email address is "guest20*XX*@server1.example com". The XX should be replaced by your two digit station number. For example, if your station number is 2, your login name is guest2002; if your station number is 12, your login name is guest2012.

6. The next two fields under "Optional Information" may be left blank but if you choose to input data the Reply-To field, it should be the same as your email address from step 5 above.

7. Select the **Forward** button.

8. You should now be on the **Receiving Mail** panel where it asks you what type of server (**Server Type**) you will be connecting to. Click the down arrow on the right side and choose "IMAP".

9. Input **`192.168.0.254`** into the **Host** block and then put **`guest20`*`XX`*** into the **Username** block, following the same instructions as earlier for the *XX*

10. Where the menu says **Use secure connection (SSL)** leave "Never" as the option of choice (in the real world we would probably want to use SSL but in the classroom we will not)

11. Leave **Authentication Type** set to "Password" and select the **Remember this password** box. Select the **Forward** button

12. Now you should be looking at the **Receiving Mail** panel. Check the box for **Automatically check for**. Leave the rest of the settings in their default state and select the **Forward** button.

Copyright © 2006 Red Hat, Inc.
All rights reserved

13. Now you should be looking at the **Sending Mail** panel. Leave the **Server Type** set to "SMTP" and enter `192.168.0.254` where it says **Host** Leave **Use secure connection (SSL)** set to "Never" and do not select **Server requires authentication**. Click the **Forward** button.

14. Now you should be looking at the **Account Management** screen. Leave this screen as it is and click the **Forward** button.

15. Now you should be looking at the **Timezone** screen. Select the closest geographical point available on the map and then select the **Forward** button.

16. Now you should be looking at the **Done** screen. Select the **Apply** button to save your configuration settings

17. At this point the Evolution application will start, and will ask you the password for guest20XX. Your password is `password`. Spend a few minutes examining the capabilities of the application. Send email to classmates. Note that your mail will arrive in the **guest20XX@server1.example.com** account, not in the Inbox under **On This Computer**.

18. Execute an **ls -aF** command in your home directory now that you have finished this exercise. Compare the output to the **ls** command output that you ran at the beginning of the exercise. Do you see a difference? What does this tell you about where evolution stores it's configuration data?

Copyright © 2006 Red Hat, Inc
All rights reserved

Challenge Sequence 5: Using digital signatures in evolution

Instructions:

1. In a terminal window type the following command and follow its instructions to create a gpg encryption key. When asked for your email address, use the guest20XX@example.com account that you configured evolution to use in the previous sequence. You will also be prompted at one point to specify a passphrase. This passphrase will be needed to sign emails or to decode encrypted emails later in the lab. Use the default values for questions about key size, algorithm, etc.

```
[student@stationX ~]$ mkdir ~/.gnupg
[student@stationX ~]$ gpg --gen-key.
```

2. Now run the following command to list the keys that gpg knows about. You should only see the key that you just created for now:

```
[student@stationX ~]$ gpg --list-keys
```

3. The output of the command above should include a line like:

```
pub 1024D/B55EBCA8 2004-12-21
```

The area after the slash and before the date is your ?key ID?; in this case, B55EBCA8. Note the value of your key ID. You will use it later in this sequence to tell evolution which key you would like to use for digitally signing emails

4. Open evolution

5. Select Tools->Settings from the dropdown menus

6. Select the account you created during the previous sequence

7. On the **Security** tab, enter the key ID that you got from **gpg**

8. Now click the **Always sign outgoing messages when using this account** checkbox

9. Choose a classmate to be your partner and send him or her an email. Make sure your partner does the same to you

10. In the **Sent Mail** folder select the email you just sent. The email should have a note at the bottom saying that it has been digitally signed and that the signature is valid. You can click the signature icon for more information

Copyright © 2006 Red Hat, Inc.
All rights reserved.

11. At this point you should have the signed email that your partner sent waiting for you in your **Inbox**. Open the email and you will see a note at the bottom saying that it has been digitally signed but the signature is invalid. This is because you do not yet have your partner's public key. This will be fixed in the next sequence

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Challenge Sequence 6: Sharing public keys, verifying signatures and encrypting email

Instructions:

1 In a terminal window, run the command:

```
[student@stationX ~]$ gpg --export -a KEY_ID > YOURNAME.pub
```

where *KEY_ID* is your key id from the previous sequence and *YOURNAME* is your name or some other unique identifier. This will create a text file that contains your public key. Anyone you give this file to can import your public key and authenticate messages that are digitally signed by you or encrypt the messages that they send to you.

2. In evolution, create a new email addressed to your partner. Click the **Attach** button and attach the file with your public key to the message. Your partner should do the same to you. This is how you will exchange public keys. Note that because anyone can send you their public key and claim that it came from someone else it is very important to only accept public keys when the key owner has been verified. The classroom environment provides face-to-face verification that your partner did indeed just send you a key, so you can trust the email.

3. After receiving a key from your partner, save the attachment to a file in your home directory and run the command:

```
[student@stationX ~]$ gpg --import PARTNERSNAME.pub
```

Where *PARTNERSNAME* is, of course, the unique name that he or she used when creating the public key file. If this command is successful then your partner's public key should be on your gpg "keyring" and should be displayed along with your own when you run **gpg --list-keys**.

4. Go back to evolution and re-open the signed email that your partner sent. The icon should now say that it has a valid signature, but that the signature is unverified. This is because you have not told gpg that you trust the associated key. In your console, run:

```
[student@stationX ~]$ gpg --list-keys
```

Note the key ID of your partner's public key. Then run the command:

```
[student@stationX ~]$ gpg --sign-key PARTNERS_KEY_ID
...output truncated...
```

Copyright © 2006 Red Hat, Inc.
All rights reserved

You will be asked to provide your private key's passphrase and to note how thoroughly you have verified the key's authenticity. Select option 2, indicating that you have casually verified that the key comes from who is says it comes from. Gpg will then use your private key to "sign" your partner's public key, indicating that you trust it to the specified extent. In a real-life situation where you and your partner's keys are stored on a public keyserver, your signature can be uploaded to the server to indicate your level of trust to anyone else who queries your partner's key. Doing this, however, is beyond the scope of this lab

5. Re-open the signed email that your partner sent once more. The signature should now be shown as valid because it comes from a signature that is both on your gpg keyring and has been signed to indicate an acceptable level of trust

6. Now that you have your partner's public key you can send encrypted email to him or her. Compose an email to your partner but before sending the email, select **Encrypt** from the **Security** menu. Make sure that your partner sends an encrypted email to you as well.

7. Check your **Inbox** for the email your partner just sent you. When you select the email you will be prompted for your private key's passphrase, which you defined in the previous sequence. After entering your passphrase, the email will be decrypted so that you can read it

Copyright © 2006 Red Hat, Inc.
All rights reserved

Unit 18

So... What Now?

1

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Objectives

Upon completion of this unit, you should:

- Understand further opportunities for exploring Linux
- Be familiar with other Red Hat Training offerings
- Know how to participate in the Linux community

2

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Some Areas to Explore

- Development
- System administration
- Further training opportunities
- The Linux Community

3

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Professional Applications of Linux

Having completed a very thorough introduction to the usage of a Linux system, students are often left wondering what they can do to further their knowledge of Linux Most users of Linux are either developers or system administrators so, although the types of professionals using Linux is expanding more and more, we will discuss each of these professions and their application to Linux.

Further Red Hat Training

We will also discuss other classes offered by Red Hat to facilitate the pursuit of either More information on Red Hat's training offerings is available at http://www.redhat.com/training.

The Linux Community

Finally we will discuss ways to get more involved in the large, exciting and vibrant community that has been built around Linux.

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Development

- RHEL includes several languages
  - Compiled Languages
    - C, C++ Java, Ada, Assembly FORTRAN 77
  - Interpreted Languages
    - Bash, Perl, Python PHP, Ruby, Lisp/Scheme
  - Programmers' Editors
    - **vi/vim**, **emacs/xemacs**, the Eclipse IDE
  - Lots more!

4

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Software Development in Red Hat Enterprise Linux

Red Hat Enterprise Linux offers a large number of programming languages. New languages are often first available on a Linux system Each programming language offers advantages in certain situations For example, FORTRAN is widely used for numeric computations and C/C++ is often the language of choice for systems applications.

Interpreted languages offer rapid program development and compiled languages offer efficient execution; Red Hat Linux provides implementations of most popular languages Additional compilers and interpreters for most languages are available on the net A list of many compilers is available at:

http://www idiom.com/free-compilers/

but this list is by no means exhaustive

Copyright © 2006 Red Hat, Inc.
All rights reserved.

Red Hat Development Classes

- RHD143: Red Hat Linux Programming Essentials
- RHD221: Red Hat Linux Device Drivers
- RHD236: Red Hat Linux Kernel Internals
- RHD256: Red Hat Linux Application Development and Porting

5

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Red Hat Developer Courses

Most classes are a 5-day combination of lecture and hands-on labs. They all provide a thorough coverage of each topic. Other Red Hat programming classes include:

RHD237 Red Hat Linux Crash Dump Analysis

RHD248 Red Hat Embedded Systems Engineering

Detailed information on Red Hat's development classes is at:
http://www.redhat.com/training/developer/courses/

Copyright © 2006 Red Hat, Inc.
All rights reserved

System Administrator Duties

- Install new systems
- Manage users and groups
- Keep software up-to-date
- Configure the network
- Configure services
- Maintain security
- Just about everything else!

6

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Further Study in System Administration

Now that you know your way around a Red Hat Enterprise Linux system you are ready to explore system administration topics. The best approach to this is to learn by doing: think of a project that would be of interest to you and set about gathering the resources that you will need to accomplish it. Further Red Hat training and use of community resources will both be of great help toward this end

Copyright © 2006 Red Hat, Inc.
All rights reserved.

RHCE/RHCT Skills Courses

- RH133: Red Hat Linux System Administration
- RH253: Red Hat Linux Networking Services & Security Administration
- RH300: Red Hat Certified Engineer Rapid Track

7

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

System Administration Courses

Red Hat offers courses that build upon the knowledge gained in RH033. Along the way students can achieve the Red Hat Certified Technician and Red Hat Certified Engineer certifications. More information on these classes is available at:

http://www.redhat.com/training/rhce/courses/

Copyright © 2006 Red Hat, Inc.
All rights reserved.

RHCA Skills Courses

- RHS333: Red Hat Enterprise Security
- RH401: Red Hat Enterprise Deployment and Systems Management
- RH423: Red Hat Enterprise Directory Services and Authentication
- RH436: Red Hat Enterprise Storage Management
- RH442: Red Hat Enterprise System Monitoring and Performance Tuning

8

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700.

Red Hat Certified Architect Courses

For post-RHCE level students Red Hat offers the Red Hat Certified Architect line of classes. These expert-level courses teach advanced topics for those seeking mastery of a particular element of Red Hat Enterprise Linux. Detailed information about RHCA courses is available at:

http://www.redhat.com/training/architect/courses/

The Linux Community

- Linux User Groups (LUGs)
- Mailing lists
- Web Sites
- IRC

9

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used copied, or distributed please email <training@redhat com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Getting involved with the community

A web search will probably reveal a Linux Users' Group (LUG) in your area. LUGs hold regular meetings to give presentations, offer support and run "Installfests" where prospective Linux users can bring in an old machine and install Linux with the assistance of LUG members. Participating in your local LUG is a great way to increase your skill and involvement with Linux. Most LUGs, distributions and software projects use mailing lists for distributing announcements and allowing users to share questions and conversation. Red Hat hosts a large number of mailing lists, which can be viewed and subscribed to at: http://www redhat.com/mailman/listinfo/. Some lists of note are:

- `redhat-list` Support for Red Hat distributions in general, RHEL in particular
- `fedora-list` Support for all versions of Fedora (very high-volume!)
- `fedora-announce-list` Announcements of security updates and new versions of Fedora

There are also many, many websites devoted to Linux support and discussion. A few places to start:

- http://www redhat.com/apps/support/knowledgebase: Red Hat's knowledge base
- http://www.tldp.org: The Linux Documentation Project
- http://www.linuxquestions.org: Home to numerous Linux message boards

You can communicate with others in real-time via Internet Relay Chat (IRC) using a client such as Xchat, which is included with Red Hat Enterprise Linux. Most IRC networks have a Linux support channel, usually called `#linuxhelp`. The Freenode network hosts channels for numerous open-source projects including the official Fedora support channel (`#fedora`) and a channel for Red Hat Enterprise Linux support (`#rhel`). As with all community support, be aware that none of these channels are necessarily populated by Red Hat employees and should not be considered a substitute for enterprise support

End of Unit 18

- Questions and Answers
- Summary
  - What to do from here?
    - Development
    - System administration
    - Community involvement
    - Further training
    - Something else? Explore!

10

For use only by a student enrolled in a Red Hat training course taught by Red Hat, Inc. or a Red Hat Certified Traning Partner. No part of this publication may be photocopied, duplicated, stored in a retrieval system, or otherwise reproduced without prior written consent of Red Hat, Inc. If you believe Red Hat training materials are being improperly used, copied, or distributed please email <training@redhat.com> or phone toll-free (USA) +1 (866) 626 2994 or +1 (919) 754 3700

Copyright © 2006 Red Hat, Inc
All rights reserved