LPI101

LPIC-1 Exam Prep (course 1)

Version: LPI101-R7S12U1404-G00

Table of Contents

Chapter Numbers & Titles

Table of Contents

Appendix

Course Logistics

Schedule:

- Class start and end times
- Lunch and other breaks

Training Facility Information:

- Restrooms
- Telephone and network access
- Break rooms and other resources
- Emergency procedures

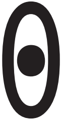
Introductions

Participant Introductions:

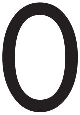
- Name and employer
- Background and relevant experience
- Objectives and topics of interest

Book Design

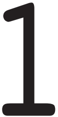
Typographic Conventions:



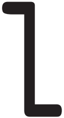
The number *"zero"*.



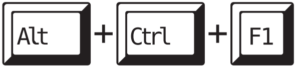
The letter *"oh"*.



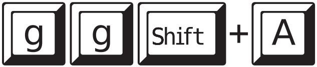
The number *"one"*.



The letter *"ell"*.



Keys pressed at the same time.



Keys pressed in sequence.

Book Design

Line Wrapping:

```
password required /lib/security/pam_cracklib.so retry=3↲
   type= minlen=12 dcredit=2 ucredit=2 lcredit=0 ocredit=2
password required /lib/security/pam_unix.so use_authtok
```

Representing File Edits:

	File: /etc/ssh/sshd_config
	#LoginGraceTime 2m
-	~~#PermitRootLogin yes~~
+	**PermitRootLogin no**
+	**AllowUsers sjansen**
	#StrictModes yes

Book Design

Command Prompts:

```
stationX$ whoami
guru
stationX$ ssh root@stationY
root@stationY's password: password
stationY# whoami
root
```

Distribution Specific Information:

```
        $ grep -i linux /etc/*-release | cut -d: -f2
[RHEL4] Red Hat Enterprise Linux ES release 4 (Nahant)
[SLES9] SUSE LINUX Enterprise Server 9 (i586)
```

Book Design

Action Lists:

Alt + F2	Open the "run" dialog.
xkill Enter	Launch `xkill`. The cursor should change, usually to a skull and crossbones.
Click on a window of the application to kill.	Indicate which process to kill by clicking on it. All of the application’s windows should disappear.

Callouts:

```
[SLES9] $ sux -
        Password: password
        # xclock
```

On SUSE, the sux command copies the MIT-MAGIC-COOKIE-1 so that graphical applications can be run after switching to another user account. The normal su command does not do this.

Chapter 1

WORK ON THE COMMAND LINE

LPI Objectives Covered

103.1 Work on the Command Line

Role of Command Shell

Shell provides user-interface

- access to filesystem
- scriptability for task automation
- program launching
- process control interface

Shells

Research Unix `/bin/sh` (original by Thompson)

- Bourne Shell (Edition 7-10 Unix)
- Rc (Plan 9, Edition 10 Unix)

C Shell (`csh`, `tcsh`)

Korn Shell

- ksh88
- Public Domain Korn Shell (**`pdksh`**)
- MirBSD Korn Shell (**`mksh`**)

  Legacy Korn shell (**`lksh`**)
- Korn Shell 93 (**`ksh`**)

Bourne-Again Shell (`bash`)

Debian Almquist Shell (`dash`)

Zsh (`zsh`)

Gathering System Info

Who else is logged into the system?

- `users`
- `who`
- `w`
- `finger`
- `last`

What type of system is this?

- `cat /etc/os-release`
- `uname -a`
- `free`

What is the system's network name

- `uname -n` and `hostname` and `cat /etc/hostname`
- `ip` and `ifconfig`

Identifying the Shell

Default login shell name is stored in the `$SHELL` environment variable

Identifying the login shell:

```
$ echo $SHELL
```

Identifying the current shell:

```
$ ps -f
```

Changing the Shell

Use the shell name to invoke that shell (i.e. type `ksh`)

Changing login shell permanently

- Edit the `/etc/passwd` entry for that user
- **`chsh`** – (change shell) available to normal users

  `/etc/shells` contains list of allowed shells

Shell Prompts

Prompt is set as hostname using the `uname -n` **command**

```
$ PS1="$(uname -n | sed 's/\..*//')$ "
homer$ export PS1
```

Prompt is set as hostname using the `net-tools` `hostname -s` **command**

```
$ PS1="$(hostname -s)$ "
homer$ export PS1
```

Bash: Bourne-Again Shell

Completely backwards compatible with Bourne shell
Adds several POSIX and TCSH enhancements to the Bourne shell

- command-line history and completion
- aliases
- sophisticated prompt configuration
- both Emacs and vi style command line editing
- tilde (~) as an alias for home directories

Navigating the Filesystem

Changing and displaying directories

- `cd`, `pwd`

Absolute vs. relative addressing

Special cases

- `cd (without parameters)`
- `cd ~`*`username`*
- `cd ~`
- `cd -`
- `.` and `..`

Help from Commands and Documentation

command `--help`

Documentation for installed packages

- RHEL7 `/usr/share/doc/`*package_name*`-`*version*
- SLES12 `/usr/share/doc/packages/`*package_name*
- U14.04 `/usr/share/doc/`*package_name*

Shipped or online distribution documentation

Linux Documentation Project - TLDP

Online help:

- web sites, FAQs, Howtos, newsgroups, mailing lists

Linux User Group(s) (LUGs)

- membership typically by mailing list subscription (no dues)
- monthly presentations/meetings

Getting Help Within the Graphical Desktop

Graphical Help begins with F1

- Gives a help manual for the active window

`yelp`

`khelpcenter`

Getting Help with man & info

It may seem cryptic, but at least it's well-documented

- `man` *[section] name*
  - man sections
  - useful options
- `info`
  - created by the GNU project
  - meant as a "superior" replacement for **`man`**
  - uses HTML like navigation with links
  - if **`info`** pages exist, they usually provide more complete and up-to-date documentation than the corresponding **`man`** page
  - use **`pinfo`** to view pages

Bash: Command Line History

View most recent commands entered

```
$ history
```

Execute previous command

```
$ !!
```

Last command starting with *xy*

```
$ !xy
```

Run command found on specified `history` line number:

```
$ !42
```

Special Control sequences can search history Ctrl+r

- see **`info bash`** for details

Fix Command may be used for advanced searching and editing:

```
$ fc -1 -5
```

Bash: Command Editing

Bash shell offers vi-mode and Emacs-mode command editing

- to set **vi** editing mode

```
    $ set -o vi
```

- to set Emacs editing mode (default)

```
    $ set -o emacs
```

Key bindings for vi-mode and emacs-mode can be easily viewed and modified

- System key bindings set in `/etc/inputrc`
- User key bindings set in `~/.inputrc`
- The Bash built-in command **bind** can be used to list and modify key bindings

Bash: Command Completion

Procedure depends on editing mode in use

- Tab for completion in Emacs mode
- Esc+\ for command completion in `vi` mode

More advanced completion than `csh` or `ksh`

- supports: command, file / directory name, username,
- hostname, and variable name completion.
- attempts to "do the right thing" based on context
- highly customizable

Shell and Environment Variables

Useful in shell scripting

Programs may malfunction if not set (`$PATH`, `$HOME`, `$USER`, etc.)

Viewing variables

- `set` (shell)
- `env` (environment)

Clearing variables

- `unset` (shell|environment)
- `env -u|i` *`command`* (environment)

Key Environment Variables

`$PATH` **– Executable search path**
`$PWD` **– Path to current working directory**
`$TERM` **– Login terminal type (e.g.** `vt100`**,** `xterm`**)**
`$SHELL` **– Path to login shell (e.g.** `/bin/sh`**)**
`$HOME` **– Path to home directory (e.g.** `/home/joe`**)**
`$USER` **– Username of user**
`$DISPLAY` **– X display name (e.g.** `station2:0.0`**)**
`$EDITOR` **– Name of default editor (e.g.** `ex`**)**
`$VISUAL` **– Name of visual editor (e.g.** `vi`**)**

Lab 1

Estimated Time:

S12:	35 minutes
R7:	35 minutes
U1404:	35 minutes

Chapter 2

USE STREAMS, PIPES, AND REDIRECTS

LPI Objectives Covered

103.4 Use Streams, Pipes, and Redirects

File Redirection

File or I/O redirection allows you to redirect STDIN, STDOUT, **and** STDERR **to files**

Requires special notation on the command line

- redirect standard input with <

```
$ sort < /etc/passwd
```

- redirect standard output with >

```
$ echo 100000 > /proc/sys/fs/file-max
```

- redirect standard error with 2>

```
$ ls -alR /proc/ 2> /dev/null
```

- redirect both STDOUT and STDERR to the same file:

```
$ ls -R /proc/ > output 2>&1
$ ls -R /proc/ &> output
```

Piping Commands Together

Piping allows the STDOUT **from one program (on the left of the pipe) to become the** STDIN **of another (on the right of the pipe)**

- The pipe symbol, **|**
- Examples:

```
$ ls -al | less
$ cut -d: -f6 /etc/passwd | sort | uniq -c | sort -rn
```

Redirection and piping can be combined:

- Usually used for feeding STDERR into the pipeline along with STDOUT:

```
# ls /proc/ 2>&1 | grep kernel
```

The tee **command writes both to STDOUT and the specified file.**

Filename Matching

Many commands take a list of filenames as arguments tedious to manually type many filenames

Wildcard patterns provide an easy way to supply many filenames as arguments

Historically called "file globbing"

Wildcard patterns are specified with special characters

File Globbing and Wildcard Patterns

A wildcard pattern is a string that contains one of the characters:

- `?` – matches any single character
- `*` – matches anything (any number of characters)
- `[...]` – character classes

  the **-** character denotes a range

  examples: `[abcd2345]` `[a-d2-5]` `[a-gA-Z0-5]`

Brace Expansion

Allows generation of arbitrary strings

Similar to wildcards, but target files or directories don't need to exist

- Can have optional preamble and/or postamble

  `{m,n,o,on}` expands to: `m`, `n`, `o` and `on`

  `d{m,n,o,on}t` expands to: `dmt`, `dnt`, `dot` & `dont`, where **`d`** is the preamble and **`t`** is the postamble
- Can also expand to a range

  e.g. `{1..1000}`, or `{a..m}`
- Can be combined with wildcards; brace expansion occurs before globbing

General Quoting Rules

Metacharacters

- Characters reserved for use by the shell

Backslash

- Do not interpret the character following as a metacharacter
- Special use of the backslash (\) to do line-wrapping or continuation

Double Quotes

- Do not interpret most enclosed characters as metacharacters
  Allows for dollar-sign expansion (e.g. variables)

Single Quotes

- Do not interpret any enclosed characters as metacharacters

Nesting Commands

Command Substitution

- Substitutes command output in place of the nested command

Nesting Commands

- `$(`*`command`*`)`
- `` ` ``*`command`*`` ` ``

Evaluating Command Output

- `eval` *`command`*

Gotchas: Maximum Command Length

Shell commands have a maximum length

Shell expansion can make a seemling-small command too big

`xargs` is designed to overcome this limitation

Gotchas

- Commands overwriting their own progress
- Whitespace and other confusing characters

Lab 2

Estimated Time:

S12:	40 minutes
R7:	40 minutes
U1404:	40 minutes

Chapter 3

MANAGE FILE PERMISSIONS AND OWNERSHIP

LPI Objectives Covered

104.5 Manage File Permissions and Ownership

104.7 File System Files and Places Files in the Correct Location (partial)

Filesystem Hierarchy Standard

Filesystem standard – FHS

- Guiding principles for each area of filesystem
- Predictable location of files and directories
  - Provides uniformity across multiple Linux distributions

The Linux Standards Base

- Aims to allow Linux binaries to run unmodified on multiple Linux distributions
- Specifies system and library interfaces and environment
- Incorporates the FHS

Displaying Directory Contents

`ls` **List directory contents**

- **`-a`** show all files (including `.`*`hidden`* files)
- **`-l`** long listings
- **`-d`** show directories not contents
- **`-h`** human readable file sizes
- **`-R`** recursively list sub-directories
- **`-S`** sort file list by size

Colored directory listings

- **`ls --color=auto`**
- `$LS_COLORS`

Filesystem Structures

Data Blocks

- The file's data.

inode Tables

- Data about the file's data.
- Note: XFS creates inodes on demand

Determining Disk Usage With df and du

`df` **Report disk space usage per filesystem**

- **`-h`** human readable output
- **`-i`** list inode information instead of block usage
- **`-T`** include filesystem type
- **`-H|--si`** use powers of 1000 instead of 1024

`du` **Report disk usage per file and directory**

- **`-h`** human readable sizes
- **`-s`** summarize, only display total for each argument
- **`-x`** do not include files on a different filesystem
- **`--si`** use powers of 1000 instead of 1024

File Ownership

Each file is owned by a specific UID and GID

`chown` **– Change the user (UID) ownership**

- Only `root` can change ownership to another user
- Can also be used to change group at the same time
- **`-R`**: change owner of directory recursively

`chgrp` **– Modify just the group (GID) ownership**

- Can only change group ownership if a member of the target group
- **`-R`**: change group owner of directory recursively

Default Group Ownership

Newly created files will usually be given GID ownership based on the current active group of the person who creates the file

`newgrp` *`newgroup`* **- log in to a new group**

- newly created files will be owned by the new group
- users can only change to their own groups
- `root` user can change to any group
- **`exit`** to switch back

File and Directory Permissions

`ls -l` List file permissions

- first character represents type of file (`d`,`-`,`l`,`b`,`c`,`s`,`p`)

Then permission sets for:

- user -UID that owns the file (sometimes called owner)
- group -GID that owns the file
- everyone else (sometimes called other)

Permissions can be represented in two ways

- symbolic representation (e.g. `rwxr-xr-x`)
- numeric representation (e.g. `0755`)

File Creation Permissions with umask

Default permissions for newly created filesystem objects

- files: `666`
- directories: `777`

umask

- defines what permissions to withhold from the default permissions
- used to display or change your umask
- usually set in the user or system shell dot files
- used to provide the user private group (UPG) scheme

Changing File Permissions

`chmod` Modify file permissions

- **`-R`** recursively modify permissions
- supports both numeric and symbolic notation
- special permissions cause different behavior for files and directories
  - set UID (SUID), set GID (SGID), sticky

File manager (GUI)

- Nautilus (GNOME)
  - Special permissions not modifiable
- Dolphin (KDE)
- Execute bit modification is all or nothing for UGO

SUID and SGID on files

The SUID bit changes the security context of an executable

An executable is normally run with the security context of the user who invoked it

An executable with the SUID bit set runs with the security context of the user who owns it, regardless of the executing user

SGID and Sticky Bit on Directories

SGID

- Files or sub-directories created within that directory inherit the group ownership of the SGID directory
- Often used to facilitate collaboration among users who need to share files

Sticky bit

- Normally in a directory that is world writable, users can delete each other's files. Setting the sticky bit overrides this behavior

User Private Group Scheme

UPG provides a convenient way to share files when working in a group project directory

UPG scheme implemented by:

1. placing each user in their own private group
2. setting the umask to `0002`
3. setting the group ownership of the project directory to a commonly shared GID
4. setting the project directory SGID

Enabling UPG on SUSE Linux Enterprise Server

- set file-creation mask to 002
- create a wrapper shell script that creates/uses private groups

Lab 3

Estimated Time:

S12:	30 minutes
R7:	30 minutes
U1404:	30 minutes

Chapter 4

CREATE, DELETE, FIND, AND DISPLAY FILES

LPI Objectives Covered

103.3 Perform Basic File Management (partial)
104.6 Create and Change Hard and Symbolic Links
104.7 Find System Files and Place Files in the Correct Location
102.3 Manage Shared Libraries

Directory Manipulation

Standard manipulation commands

- `mkdir` – creates directories
  - `-m`: set permissions on new directory
  - `-p`: Create parent directories if they don't exist
- `rmdir` – deletes empty directories
  - `-p`: Remove empty parent directories

File Manipulation

Standard manipulation commands

- **`cp`** – copies files and directories

  `-a`: Archive recursively, preserving permissions, ownership, links and not following symbolic links, etc.

  `-R|-r`: copy directories recursively
- **`mv`** – moves or renames files and directories

  `-u`: Overwrite only if destination is older than source
- Shared options

  `-f`: replace file without prompting (see **`-i`**)

  `-i`: prompt before replacing a file

Deleting and Creating Files

`rm` **– removes (deletes) files and directories**

- **`-i`**: prompt before removing
- **`-f`**: do not prompt before removing
- **`-R|-r`**: remove directories recursively (including contents)

`touch` **– creates empty files or updates** `mtime` **and** `atime` **on existing files**

- **`-a`**: Set only atime to the current time
- **`-m`**: Set only mtime to the current time
- **`-t`**: Set both atime and mtime to a specified time

Physical Unix File Structure

Block and inode based

- blocks hold data
- inodes hold metadata

Superblock contains filesystem parameters

- How many inodes, etc

Filesystem Links

Created with `ln`

Hard links – directory entry that references the same inode as another directory entry

- can't span filesystems
- can't create hard links to non-existent file
- can't create hard links to directories
- do not require additional storage space (i.e. blocks)

Symbolic links – file that references another file via path and name

- can reference directories
- can span filesystems
- can reference non-existent files
- occupy space

File Extensions and Content

File extensions have no special meaning to the kernel

- file extensions are just part of the file name
- the kernel only distinguishes between executable and non-executable (data) files
- some applications may care about extensions or otherwise use them for user convenience features

  Apache **`httpd`**, **`tar`**, **`gzip`**, file managers like Midnight Commander, OpenOffice/LibreOffice, Calligra

The `file` command reports the type of file by examining the file contents

Which and Type

`which`

- Prints the absolute path of the first matching command in `$PATH`.

  Especially useful if there is more than one program of the same name (e.g. `/bin/ls`, `/opt/bin/ls`).
- **`-a`**

  Prints the absolute path of all matching commands

`type`

- Identifies the type of command being executed (e.g. shell script, binary, alias).

whereis

`whereis`

- Used to identify the location of a command, related source code, and related man pages

  `-m` | **`-M`**: restricts results to manual page(s)
  `-b` | **`-B`**: restricts results to executable files

Searching the Filesystem

`find` – searches a directory structure for requested files

- First argument(s) are path(s) to start search from; default is current directory
- Next arguments specify criteria to search on: file name, size, permissions, type, owner, group, atime, mtime, ctime
- Last argument specifies action to perform.

  `-print` is the default action and displays matches

  `-ls` displays full details on matches

  `-exec` allows a command to be run against each matching file. The **`-ok`** can be used when a confirmation prompt is desired

Alternate Search Method

`locate` **– High-speed and low-impact searching**

- Searches index instead of actual filesystem
- Index updated each night by default

  `locate` won't know about recently added/deleted files until database is rebuilt
- Search criteria limited to pattern matching in the pathname and filename

Manually Installed Shared Libraries

Check for library dependencies with the `ldd` command

`ld-linux.so` **locates libraries to link to**

- `/etc/ld.so.conf`

  `/etc/ld.so.conf.d/*.conf`
- `/etc/ld.so.cache`

Run `ldconfig` command after installation of new libraries

Lab 4

Estimated Time:

S12:	20 minutes
R7:	20 minutes
U1404:	20 minutes

Chapter 5

WORK WITH ARCHIVES AND COMPRESSION

LPI Objectives Covered

103.3 Perform Basic File Management (partial)

Archives with tar

`tar`/`star`

- manipulates `.tar` files, also called tarballs
- used for backup and transfer of files
- creates, extracts or lists the contents of tarballs

`.tar` **(tarball)**

- records file and directory structure
- includes metadata about the file: date, timestamps, ownership, permissions, etc.

Compression/Decompression options

- compress, gzip, bzip2, lzma/xz

Archives with cpio

Features of `cpio` archives include:

- manipulates `.cpio` files
- used as the basis for RPM packages
- doesn't recurse sub-directories, must be passed list of dirs
- more robust than **`tar`** when media errors encountered
- **`-i`** → input mode, used when feeding a cpio archive into the **`cpio`** command
- **`-o`** → output mode, used to create cpio archives, which are sent to `STDOUT`

The gzip Compression Utility

`gzip` – popular replacement for `compress`

- created by the GNU project because of patented algorithms in **`compress`**
- default action deletes original after creating new compressed file
- standard file extension: `.gz`
- much higher compression ratio than **`compress`**

`gunzip` or `zcat` decompresses files compressed with `gzip`

- **`gunzip`** decompresses the file on disk (removing the original, compressed file); **`zcat`** does not
- **`zcat`** outputs uncompressed data to `STDOUT`

The bzip2 Compression Utility

bzip2

- typically better compression than the **gzip** command
- default action deletes original after creating new compressed file
- standard file extension: `.bz2`

bunzip2 or bzcat decompresses files compressed with bzip2

- **bunzip2** decompresses the file on disk (removing the original, compressed file); **bzcat** does not
- **bzcat** outputs uncompressed data to `STDOUT`

The XZ Compression Utility

`xz` Latest and greatest compression

- better compression than the **`bzip2`** command
- default action removes original file after creating new compressed file
- standard file extension: `.xz`
- legacy file extension: `.lzma`

  Use **`--format=lzma`** for LZMA support

`xz -d` (`unxz`) or `xz -dc` (`xzcat`) decompresses files compressed with `xz`

- **`xz -d`** decompresses the file to disk (removing the original, compressed file); **`xz -dc`** does not
- **`xz -dc`** prints uncompressed data to `STDOUT`

Replaces `gzip` and `bzip2` as compression format of choice

The PKZIP Archiving/Compression format

`zip` – Compatible with PKZIP files

- default action does NOT delete original file(s) after creating new compressed archive
- standard file extension: `.zip`

`unzip` expands a `.zip` file

Unlike other Linux compression formats, the PKZIP format can store multiple files and directories natively

Lab 5

Estimated Time:

S12:	25 minutes
R7:	25 minutes
U1404:	25 minutes

Chapter 6

PROCESS TEXT STREAMS USING FILTERS

LPI Objectives Covered

103.2 Process Text Streams Using Filters

Producing File Statistics

`wc` – Counts lines, words, characters and bytes in text files

- when given multiple files as arguments, produces totals for each file as well as an overall total
- can be told to only output total for lines, words, characters, or bytes
- most common usage is to count lines

The Streaming Editor

`sed` – A [s]treaming [ed]itor

- performs edits on a stream of text (usually the output of another program)
- often used to automate edits on many files quickly
- small and very efficient
- Standard Options

  `-n`

  `-e`

  `-f`
- Useful GNU extensions

  `-r` option for extended regular expression use

  `-i` option for in place edits with modern versions

Replacing Text Characters

`tr` – translates, squeezes & deletes characters

- translates one set of characters into another

  commonly used to convert lower case into upper case

  ```
  tr a-z A-Z
  ```
- squeeze collapses duplicate characters

  commonly used to merge multiple blank lines into one

  ```
  tr -s '\n'
  ```
- deletes a set of characters

  commonly used to delete special characters

  ```
  tr -d '\000'
  ```

Text Sorting

`sort` – Sorts text

- can sort on different columns
- by default sorts in lexicographical order

  1, 2, 234, 265, 29, 3, 4, 5
- can be told to sort numerically (by using the **`-n`** option)

  1, 2, 3, 4, 5, 29, 234, 265
- can merge and sort multiple files simultaneously
- can sort in reverse order
- often used to prepare input for the **`uniq`** command

Duplicate Removal Utility

`uniq` **– Removes duplicate adjacent lines from sorted text**

- cleanly combines lists of overlapping but not identical information
- **`-c`** prefixes each line of output with a number indicating number of occurrences
- taking this output and performing a reverse sort produces a sorted list based on number of occurrences

Extracting Columns of Text

`cut` **– Extracts selected fields from a line of text**

- can specify which fields you want to extract
- uses tabs as default delimiter
- **`-d`** option to specify a different delimiter
- most useful on structured input (text with columns)

Displaying Files

`cat` – **displays entire file(s)**
`nl` – **displays entire file(s) with line numbers added**
`more` – **displays file(s) one screen at a time**
`less` – **more sophisticated and configurable pager**

Prepare Text for Display

`pr`

`expand` / `unexpand`

Previewing Files

`head` **– displays first 10 (by default) lines of file**

`tail` **– displays last 10 (by default) lines of file**

- **`tail -f`** to watch a file be appended to

Use the `-n` option to configure how many lines to view

Displaying Binary Files

Displaying raw binary data may corrupt the display terminal

- `reset` corrects terminal
- Ctrl+j`reset`Ctrl+j (if typing Enter fails)

`strings` **– displays ASCII text inside binary files**

`od` **– displays a dump of a file in various formats**

`xxd` **– displays HEX and ASCII dump of file**

Combining Files and Merging Text

`cat` – Concatenate files
`paste` – Merges text from multiple files

- **`-s`** option to merge files serially
- uses tabs as default delimiter

Lab 6

Estimated Time:

S12:	15 minutes
R7:	15 minutes
U1404:	15 minutes

Chapter 7

SEARCH TEXT FILES USING REGULAR EXPRESSIONS

LPI Objectives Covered

103.7 Search Text Files Using Regular Expressions

Searching Inside Files

`grep` – **searches for patterns within files**

- **`-A`** ***`NUM`*** ⇒ print match and *`NUM`* lines after match
- **`-B`** ***`NUM`*** ⇒ print match and preceding *`NUM`* lines
- **`-C`** ***`NUM`*** ⇒ print match and *`NUM`* lines before and after
- **`-E`** ⇒ use extended regular expressions
- **`-F`** ⇒ match fixed strings, not regular expression
- **`-i`** ⇒ perform case insensitive match
- **`-l`** ⇒ print name of file(s) containing a matching line
- **`-n`** ⇒ show line numbers
- **`-R`** ⇒ recursively descend through directories
- **`-v`** ⇒ invert match; prints what doesn't match
- **`--color`** ⇒ highlight matched string(s) in color

Regular Expression Overview

Regular Expressions (REs) provide a mechanism to select specific strings from one or more lines of text

- Rich and expressive language
- Used by many commands and programming languages:
  `grep`, **`awk`**, **`sed`**, Emacs, **`vi`**, **`less`**, Expect, **`lex`**, Perl,
  Python, Tcl, Delphi, and Microsoft Visual C++

Regular Expressions

The building blocks of regular expressions are expressions that match a single character

- most characters, letters and numbers match themselves
- special characters are matchable as well
- "." (the period) matches any single character
- specify where the match must occur with anchors

RE Character Classes

Character classes, *[...]*, match any single character in the list

- RE `[0123456789]` matches any single digit

Some predefined character classes

- `[:alnum:]` `[:alpha:]` `[:cntrl:]` `[:digit:]`
- `[:lower:]` `[:punct:]` `[:space:]` `[:upper:]`

The - character denotes a range

RE `[[:alnum:]]` equivalent to `[0-9A-Za-z]`

- Matches any single letter or number character

Regex Quantifiers

Interval expressions (within curly braces)

- `{42}` → match exactly 42 times
- `{42,}` → match at least 42 times
- `{0,42}` → match no more than 42 times or none

The three most commonly needed quantifiers have single-character abbreviations:

- `*` matches 0 or more and is equivalent to `{0,}`
- `+` matches 1 or more and is equivalent to `{1,}`
- `?` matches 0 or 1 and is equivalent to `{0,1}`

RE Parenthesis

Parenthesis

- `(`***`RE`***`)` → creating a new atom
- `(`***`RE`***`)\`***`non-zero digit`*** → storing values
- `(`***`RE1`***`|`***`RE2`***`)` → alternation: ***`RE1`*** or ***`RE2`***

Lab 7

Estimated Time:

S12:	35 minutes
R7:	35 minutes
U1404:	35 minutes

Chapter 8

PERFORM BASIC FILE EDITING OPERATIONS USING VI

LPI Objectives Covered

103.8 Perform Basic File Editing Operations Using vi

Text Editing

Unix Revolves Around Text

- Text is robust
- Text is universally understood
- The only tool / program required is a text editor
- Remote administration possible over low-bandwidth connections

Text Editors

- Many editors available, each with fanatical followings
- Pico/Nano, vi and Emacs are the most common
- `$EDITOR` and `$VISUAL` control default editor

vi and Vim

`vi` **– Visual Display Editor**

- Developed originally by Bill Joy for BSD
- Available on all Unix platforms as a POSIX standard

Vim – Vi IMproved

- Developed originally as a clone for the Amiga
- Has significantly enhanced functionality
- Includes a POSIX compatibility mode
- Most common **`vi`** on Linux and OS X

Learning Vim

Getting help

- Friends & Co-workers
- Books & Cheat Sheets
- `:help` – Vim has extensive help documentation
- http://www.vim.org/

Basic vi

`vi` is Modal

- Insert Mode: keystrokes are inserted into the document
- Command Mode: keystrokes are interpreted as commands

Basic Cursor Movement Commands

- h j k l

Basic Editing Commands

- i a Esc x dd

Saving & Exiting

- :w
- :q
- :q!
- :x
- Shift+Z Shift+Z

Intermediate vi

Repeating Actions
Undoing Changes
Insert & Substitute
Search & Replace
Delete, Yank, & Put
More Movement Commands

Lab 8

Estimated Time:

S12:	60 minutes
R7:	60 minutes
U1404:	60 minutes

Chapter 9

CREATE, MONITOR, AND KILL PROCESSES

LPI Objectives Covered

103.5 Create, Monitor and Kill Processes
103.6 Modify Process Execution Priorities

What is a Process?

A process is a launched program
Associated with a process:

- process ID (PID)
- priority
- nice value
- memory
- security context
- environment
- file handles
- exit status

Process Lifecycle

Processes are organized in a hierarchy

- **`init`** – first process spawned by kernel with PID of 1
  the only process directly launched by the kernel
  `init` will spawn child processes
- child processes spawn other children, etc.

Processes can be created by two methods

- `fork()` – create child duplicate of self
- `exec()` – spawn completely new process that replaces parent
- `fork()` + `exec()` – method for launching different process

Process termination methods

- Normal termination via `exit()`
- Abnormal termination via `abort()` or uncaught signal

Process States

Processes can transition between states upon receipt of signals

- running ⇒ currently being allocated CPU slices
- stopped ⇒ still loaded in memory, but not running
- sleeping ⇒ waiting for some event (ex. user input)
- un-interruptible sleep ⇒ as the name suggests; usually caused when waiting for I/O
- zombie ⇒ a terminated process whose resources have all been freed except for a PID and exit status

Viewing Processes

`ps` – **standard command to view process info**

- supports many options to modify output
- can emulate behavior of other Unix-family **`ps`** commands
- reads information from the `/proc/` filesystem

`jobs` – **display interactive, backgrounded/suspended processes**

`top` – **similar to `ps`, but interactive**

- Provides summary information and stats on each running process in real-time fashion (refreshes display every 3 seconds by default)
- can sort processes by various criteria such as CPU usage, memory, UID, etc.
- can send signals to processes

`gnome-system-monitor` – **limited GUI `top`-like program**

Signals

Special message that can be sent to a process

Processes can install signal handlers that catch signals and trigger some action

Signals can have different meanings on different architectures

Some signals cannot be caught or ignored and are processed by the kernel (e.g. `SIGKILL (9)`)

Tools to Send Signals

`kill` – Send arbitrary signals to process by PID

- sends a `SIGTERM` (15) by default
- **`-l`** lists all signals supported on the machine

`killall` – Send signal to process by name

`pkill` – Send signal to process by terminal, group, username, PID, or command name

`top`, `gnome-system-monitor`, `ksysguard` can also send signals

Certain key bindings send signals

- Ctrl+C = `SIGINT` (2)
- Ctrl+Z = `SIGSTOP` (19)

Managing Processes

`kill`

- Send signals (default `TERM`) to processes

`killall`

- Sends signals to processes by name

`pgrep`, `pkill`

- Manage processes by terminal, group, username, PID, or command name
- Available for both Linux and Unix

Tuning Process Scheduling

Launch a process with an adjusted CPU scheduling priority:

- `nice` *myprogram*
- `nice -n 15` *myprogram*
- `nice -n -10` *myprogram*

Modify the CPU scheduling priority of a running process:

- `renice 10` *pid*
- `renice -10` *pid*
- `renice 0 -p` *pid*
- `snice` by terminal, group, username, PID, or command name

Advanced: Setting and viewing CPU scheduler and priority

- `chrt`

Setting and viewing I/O scheduler and priority — `ionice`

Job Control Overview

Job control gives you the ability to do multitasking at the command line

Job control refers to the ability to selectively stop (suspend) the execution of processes and continue (resume) their execution at a later point

These functions are exposed to the user via the shell

- Older or minimalist shells may not support job control

Job Control Commands

Start a process as a background process by running *`program`* `&`

Stop an already running process by sending it a `SIGSTOP (19)`**, (ex. pressing** Ctrl+Z**)**

- **`fg`** – run the job in the foreground
- **`bg`** – run the job in the background
- **`kill`** – terminate the job

Refer to jobs using `%`*n***, where** *n* **is the job number**

The `jobs` **command will list all jobs present on the shell but can not list jobs for other shells**

nohup and disown

Lost serial connections cause `SIGHUP` signal to be generated

- Processes can choose to terminate or reload
- **`nohup`** protects processes from receiving `SIGHUP`
  - Processes are not automatically backgrounded
- **`disown`** informs the shell a job should not be sent a `SIGHUP` signal.

uptime

`uptime`

- Shows the following information:

  Current time
  How long the system has been running
  The number of users logged in to the system
  The system load average.
- Reads data from `/proc/` and `/var/log/wtmp`

Persistent Shell Sessions with Screen

Terminal Multiplexer (window manager)
Allows for very efficient multitasking from a virtual terminal
Sessions can be disconnected and reconnected at will
Useful for remote administration

Using screen

Starting screen
Commands
Detaching and re-attaching to sessions
Session basics

Advanced Screen

Session locking
Split-screen
Monitoring sessions
Sharing screen sessions
Default settings

- System-wide: `/etc/screenrc`
- Per user: `~/.screenrc`

Lab 9

Estimated Time:

S12:	45 minutes
R7:	45 minutes
U1404:	45 minutes

Chapter 10

USE RPM, YUM, AND DEBIAN PACKAGE MANAGEMENT

LPI Objectives Covered

102.4 Use Debian Package Management
102.5 Use RPM and YUM Package Management

Managing Software

Unix Packaging

- SysV packages

Linux Packaging

- Slackware Tarballs
- Debian dpkg
- Red Hat RPMs

RPM Architecture

RPM package files

RPM database

- `/var/lib/rpm/`

RPM utilities

- **`rpm`**
- **`rpmquery`**, **`rpmverify`**, **`rpmsign`**
- **`rpmbuild`**
- **`rpm2cpio`**

RPM configuration files

- macros used during preparation and installation of RPMs

Working With RPMs

Installing without upgrading RPMs

- `rpm -i`

Upgrading/Installing & Freshening RPMs

- `-U` and `-F`
- `--oldpackage`, `--replacepkgs`, and `--force`

Uninstalling RPMs

- `rpm -e`

Useful Options

- `-v`, `-h`
- `--nodeps` ignore dependencies (not recommended)
- `--test` check for possible errors, but don't actually do anything

Querying and Verifying with RPM

`rpm -q | rpmquery` has powerful query capabilities:

- files – to see information about the package that supplied the file
- non-installed package files – to see package contents
- system's RPM database – to see information on installed packages

`rpm -V | rpmverify`: verify file integrity and changes

RPM GnuPG key checking

- **`rpm --import`** — Import GPG (GNU Privacy Guard) Key
- **`rpm -qa 'gpg-pubkey*'`** - List Imported Keys
- **`rpm -K`** ***package*** - Check Signature

Installing Debian Packages

The `dpkg` program processes Debian's packages

- Debian packages end in `.deb`

Querying and Verifying with dpkg

`dpkg-query` and `dpkg-deb`; main query tools for querying the dpkg database:

- **`dpkg-query`** - querying the **`dpkg`** database
- **`dpkg-deb`** - querying local Debian packages

`debsums` **- Track MD5 sums of installed packages**

The alien Package Conversion Tool

`alien` can convert package files from one format to another

- From either `.rpm` or `.deb`
- To either `.rpm`, `.deb`, or tarball (`.tar`, `.tar.gz`, `.tar.bz2`)

`alien` should only be used to convert .deb packages to tarballs

- meta-data & dependency problems make packages less reliable

`rpm2cpio` can be used for RPMs

Managing Software Dependencies

Software Management Problems

- Large dependency trees are difficult to manage
- Many applications have many dependencies

Package Management Solutions

- Uses a central repository of packages
- Inter-dependencies are automatically calculated/managed

Bundled with Red Hat Enterprise Linux

- Provided by the **`yum`** command

Bundled with SUSE Linux Enterprise Server

- Provided by the **`zypper`** command

Bundled with Ubuntu

- Provided by `APT`

Using the Yum command

Yum Package (un)installation:

- `install/localinstall`
- `reinstall`
- `update`
- `downgrade`
- `remove`

Yum Package Querying

- `info`
- `list`
- `search`
- `whatprovides`

Yum Maintenance

- `clean`

yumdownloader

`yumdownloader`

- Download one or more RPMs from YUM repositories
- Shows the URL, or URLs, of a package's YUM repository

Configuring Yum

Yum configuration

- Main configuration
  `/etc/yum.conf`
  Ubuntu: `/etc/yum/yum.conf`
- Yum repositories
  `/etc/yum.repos.d/*.repo`
- **`yum-config-manager`**
  Non-interactive repo management

The dselect & APT Frontends to dpkg

APT

- **`apt-get`** → automates downloading and installing packages and their dependencies

```
# apt-get install package_name
```

- **`apt-cache`** → search configured archives and display information for packages

```
# apt-cache show package_name
# apt-cache search expression
```

`dselect` → ncurses **interface**

Aptitude

The `aptitude` utility offers both command line and `ncurses` interfaces

- Replacement for the old **`dselect`** command
- Alternative front-end to the commonly used APT tools (e.g. **`apt-get`**)
- Searches are limited to package names; use **`apt-cache`** for better searching support
- Automatically installed packages can be automatically uninstalled

Configuring APT

APT configuration

- Main configuration

  ```
  /etc/apt/apt.conf
  /etc/apt/apt.conf.d/*
  ```

- APT archives

  ```
  /etc/apt/sources.list
  /etc/apt/sources.list.d/*.list
  ```

Keeping a Debian-based System Current

Lab 10

Estimated Time:

S12:	5 minutes
R7:	5 minutes
U1404:	5 minutes

Chapter 11

WORK WITH PARTITIONS, FILESYSTEMS, AND DISK QUOTAS

LPI Objectives Covered

102.1 Design Hard Disk Layout
104.1 Create Partitions and Filesystems
104.2 Maintain the Integrity of Filesystems
104.3 Control Mounting and Unmounting of Filesystems
104.4 Manage Disk Quotas

Partition Considerations

MBR Table Structure

- Primary Partitions (max of 4)
- Extended Partition (max of 1)
  - generally fills rest of disk
  - contains Logical Partitions
- Max number of partitions limited by kernel and partitioning tools
- 32bit LBA limits max disk size to 2TB

GPT Table Structure

- 128 partitions
- No extended or logical partitions
- Critical structures duplicated and CRC checked
- 64bit LBA limits max disk size to 9.4 billion TB

Logical Volume Management

Hierarchy of Concepts

- Physical Volumes
- Physical Extents
- Volume Groups
- Logical Volumes
- Logical Extents
- Filesystems

RHEL7: All filesystems except `/boot/` **can live on LVM**

- Anaconda limitation

Filesystem Planning

Appropriate filesystem layout depends on machine function

- Only a root filesystem (/) is absolutely required
- Typical minimum partitions: `/boot/`, `/`, and swap.

Common additional filesystems

- `/var/` – This directory contains logs, mail files and other various data
- `/srv/` – A blank directory that can be used much like `/var/`
- `/tmp/` – Space for temporary files
- `/home/` – Users' home directories
- `/opt/` – Additional program binaries (usually third party)
- `/usr/local/` – Additional local programs and data

Partitioning Disks with fdisk & gdisk

`fdisk/gdisk`

- Edits in memory copy. Only writes to disk when instructed
- **`fdisk`** — Can create and edit DOS/MBR style partition tables
  DOS/MBR created in 1983 supporting 2TB max disk
- **`gdisk`** — same UI as **`fdisk`** but works with GPT partition tables
  Can convert MBR partition tables into GPT tables
- Doesn't always update kernel structures
  use **`partprobe`**, **`partx`**, or **`addpart`** and **`delpart`**
  May require a reboot

`sfdisk/sgdisk`

- Non-interactive, Script-able via command line options/params

Resizing a GPT Partition with gdisk

Scenario: A LUN has been grown on a RAID controller or SAN

- The LUN is using a GPT partition table
- Grow the last partition to include the new space

Perform SCSI rescan so that kernel sees new block device size

Relocate the GPT data structures to the (new) end of the disk

Delete the last partition after noting:

- type, name and starting location

Create new partition with same starting location

- Ending location at the end of available space

Set partition type and name if needed

Save and reboot

Partitioning Disks with parted

`parted`

- Supports many partition table types (including MBR and GPT)
- Writes changes to disk immediately!
- Can move and resize partitions and filesystems
- Graphical version: **`gparted`**

  Uses `libparted`

Filesystem Creation

Filesystem creation with `mkfs -t` *`filesystem_type`*

- Runs commands specific to the filesystem type requested: **`mkfs.ext4`**, **`mkfs.ext3`**, **`mkfs.xfs`**, **`mkfs.btrfs`**, **`mkfs.msdos`**
- `/etc/mke2fs.conf` sets defaults for **`mke2fs`**

Filesystem Support

Support for dozens of filesystem types including:

- Minix, ext2, MS-DOS, UMSDOS, VFAT, NTFS, NFS, ISO9660, HPFS, SYSV, SMB, CIFS, OCFS2, GFS2, AFFS, BeFS, BFS, HFS, EFS, NWFS, QNX, RFS, UDF, UFS

Linux advanced logging / journaling filesystems:

- ReiserFS, JFS, ext3, ext4, XFS, Btrfs

Unix/Linux Filesystem Features

Standard Unix filesystem characteristics

- singly rooted
- cAsE SensiTiviTY
- long file names
- supports links
- timestamps various file operations
  `ctime`, `atime`, `mtime`

Swap

Swap partitions

- Type `0x82`

Swap files

- Must be contiguous

`mkswap`

`swapon`

Selecting a Filesystem

Linux supports several advanced journaling filesystems

Extended Filesystem: Ext2 (no journal), Ext3, and Ext4

- R7: Max supported filesystem size — 50TB
- R7: Max supported individual file size — 16TB

XFS – SGI's journaling filesystem

- Advanced filesystem, highly scalable, supports defragging
- Sparse inode chunk allocation to support peer-to-peer cluster filesytems: GlusterFS and Ceph
- R7: Max supported filesystem size — 500TB

Btrfs

- Built-in Volume Manager, RAID, compression, and many features
- Likely future default Linux filesystem

Filesystem Maintenance

Viewing filesystem meta-data

- XFS `xfs_info` & Ext2/3/4 `dumpe2fs`

Reconfiguring filesystem

- Ext2/3/4 `tune2fs` highlights

  set default mount options

  set auto `fsck` interval and mount count

  filesystem conversion: ext2 > ext3 or ext3 > ext4
- XFS `xfs_admin` — Parameter adjustment
- XFS `xfs_fsr` — Online Filesystem Defragger

Correcting minor filesystem problems

- `fsck` & `e2fsck`/`xfs_repair`
- Systemd boot parameter `fsck.mode=force`

Mounting Filesystems

`mount`

- **`-o`**
- **`--bind`**

`umount`

`/etc/fstab`

- List of mounts and options
- Used during boot

What is mounted or in use?

- `/etc/mtab` → symlink to `/proc/self/mounts`
- `/proc/mounts` → state recorded by kernel
- **`fuser`** → processes using a mount
- **`lsof`** → files open on a mount

Mounting Filesystems

Mount filesystems with the mount command

- **`mount`** ***[-t type] [-o option[,option[,...]]] [device] [dir]***
- searches the `/etc/fstab` file for missing parameters if supplied with only *device* or *dir*
- **`mount`** without parameters to list currently mounted filesystems

Unmount filesystems not currently in use with

- **`umount`** ***[device|dir]***

Managing an XFS Filesystem

Maintenance

- `xfs_admin`
- `xfs_repair`
- `xfs_growfs`
- `xfs_info`

Backup and Restore

- `xfs_freeze`
- `xfs_copy`
- `xfsdump`
- `xfsrestore`

NFS

The Network Filesystem is the native Unix file-sharing method

- Developed by Sun Microsystems
- NFS servers export directories
- Client machines mount NFS exports and local applications and users access files as if they were local
- Default settings are conservative; can be tuned for much higher performance

SMB

SMB is the native file sharing protocol on Microsoft Windows and many other platforms

- Developed by IBM originally
- SMB is synonymous with CIFS
- Servers share directories, printers, users and other information
- Client machines can browse shared files and printers, accessing them just like local resources

Two Linux clients

- `smbclient`
- Mount `cifs` network shares

Filesystem Table (/etc/fstab)

Contains information about filesystems

Which filesystems to mount and when

- Order is significant
- One filesystem per line

Options for mounting each filesystem

Used to mount filesystems at boot time (`auto` **vs.** `noauto`**)**

Requires `root` **access unless the** `user` **or** `users` **options are used**

Configuring Disk Quotas

Mount options for filesystems

- `usrquota`
- `grpquota`

Database records

- XFS treats quotas as filesystem metadata
- Created and populated with initial values by `quotacheck` for other file systems (e.g. ext3)

  Files used at the root of the filesystem: `aquota.user` and `aquota.group`
- Kernel updates values as disk writes occur

Starting and stopping accounting

- `quotaon`, `quotaoff`

Setting Quotas

Types of Quotas

- User Quotas
- Group Quotas

Types of Limits

- File Limits (soft or hard)
- Block Limits (soft or hard)
- Grace Period

`setquota`

- Non-interactive

`edquota`

- Interactive with text editor

XFS also uses `xfs_quota(8)`**.**

Viewing and Monitoring Quotas

Determining usage

- `quota`
- `repquota`

Notifying users

- `warnquota`

Lab 11

Estimated Time:

S12:	20 minutes
R7:	20 minutes
U1404:	20 minutes

Chapter 12

LINUX BOOT PROCESS

LPI Objectives Covered

101.2 Boot the system
102.2 Install a boot manager
101.3 Change runlevels and shutdown or reboot system

Booting Linux on PCs

Booting is a critical and complex sequence

- Linux can have very infrequent reboots (long uptimes)
- Little opportunity for SysAdmin to become familiarized
- Familiarity required for troubleshooting bootup errors

Main Actors

- System BIOS or UEFI
- Sector 0 of boot device (BIOS) or UEFI boot manager shim
- GRand Unified Bootloader (GRUB)
- Initial ramdisk
- Linux kernel
- **`/sbin/init`** launches bootup scripts from `/etc/`

GRUB 2

GRUB rewrite with many new features
Provides a menu of all available kernels which can be booted
Can interactively find kernels to boot

- Filesystem support: ext{2,3,4}, XFS, Btrfs, JFS, ReiserFS, VFAT, and many others via modules
- Hardware support: Uses the BIOS or UEFI for I/O

Can also boot other OSes for multi-boot setups
Provides support for serial consoles
Can pass options at boot prompt
RHEL7/SLES12: use `2` in the GRUB command and file names

- e.g. **`grub-install`** vs **`grub2-install`**

GRUB 2 Configuration

Configuration file:

- BIOS/MBR boot → `/boot/grub2/grub.cfg`
- UEFI/GPT boot → `/boot/efi/EFI/`*`distro`*`/grub.cfg`

`grub.cfg` **is built by `grub2-mkconfig` that runs**

- `/etc/grub.d/*` scripts which
- source `/etc/default/grub`

U14.04: `update-grub` calls `grub-mkconfig`

`GRUB_DEFAULT` **— default menuentry to boot**

- `GRUB_DEFAULT=0` — by position
- `GRUB_DEFAULT='Linux (3.17.2)'` — by name
- `GRUB_DEFAULT=saved'` — by the `saved_entry` variable

GRUB Legacy Configuration

Provides a menu of all available kernels which can be booted

Can interactively find kernels to boot

- Filesystem support: ext{2,3,4}, XFS, JFS, ReiserFS, VFAT and many others via stage1.5 drivers
- Hardware support: Use the BIOS for I/O via the int13 hook

Can also boot other OSes for multi-boot setups

Provides support for serial consoles

Can pass options at boot prompt

Configuration file:

- RHEL7 → `/boot/grub/grub.conf`
- SLES12 → `/boot/grub/menu.lst`

Boot Parameters

Passed on kernel command line

- Persistently defined in GRUB configuration file

Viewable after boot via `/proc/cmdline`

Four types of parameters

- kernel parameters - (e.g. `console=ttyUSB0`)
- initramfs dracut parameters - (e.g. `rd.debug`)
- **`systemd`** parameters - (e.g. `systemd.unit=multi-user.target`)
- Systemd unit parameters - (e.g. `fsck.mode=force`)

Kernel parameter documentation: `bootparam(7)`

Dracut documentation: (e.g. `dracut.cmdline(7)`**)**

Systemd boot parameters documentation: (e.g. `kernel-command-line(7)`**)**

init

First userspace process launched by kernel

- PID 1
- All other processes are children of init
- Troubleshooting Tip: Pass `init=/bin/sh` to launch /bin/sh instead of **`init`**

Modern Linux distributions have standardized on `systemd`

- On boot, systemd recursively activates all units that are dependencies of the `default.target`

  `default.target` is usually aliased to `multi-user.target` or `graphical.target`

`local-fs.target`→`sysinit.target`→`basic.target`→`multi-user.target` (optionally) →`graphical.target`

Linux Runlevels Aliases

Maintenance runlevels: runlevel `1`, `s` (or `single`)

- Booting into maintenance mode: Use runlevel `s` (or `1`)
- Switching running system into maintenance mode: Use only `1`

Historically the default runlevel specified in `/etc/inittab`

- Commonly, the default runlevel is either:
  - `3` – Multi-user with networking
  - `5` – Same as 3, but adds a GUI

Systemd respects these runlevel numbers via aliases

Systemd local-fs.target and sysinit.target

R7: Replaces `/etc/rc.sysinit`
S12: Replaces `/etc/init.d/boot`

`local-fs.target` **first target after initramfs, pulls in units that:**

- Checks/remounts /, Activates swap, Updates/activates disk quotas and DM RAID (if in use)
- Dynamically generate `mount` targets for each regular filesystem in `/etc/fstab` via **`systemd-fstab-generator`**

`sysinit.target` **next major target processed during boot**

- Starts **`systemd-udevd`**
- Configures kernel parameters: `/etc/sysctl.conf` & `/etc/sysctl.d/`
- Launches plymouth for graphical boot

Runlevel Implementation

`rc`

- Called by **`init`** to change runlevels, (see `/etc/inittab`)

`/etc/init.d/`

- Contains scripts to control system processes
- Scripts are called when entering, or leaving, a runlevel, and directly
- Runlevel scripts are typically added during package installation, or by system administrator

`rc#.d/` **directories**

- Contain symbolic links to the scripts in `/etc/init.d/`
- Filenames take the form `S##`*`script`* (start) and `K##`*`script`* (kill)
  - `##` indicate numeric order in which scripts are run

System Boot Method Overview

Runlevel-driven - AT&T System V init

- Each runlevel can have a unique defined list of services to `start` and `stop`
- Used by most commercial Unix systems and Linux distributions

Event-driven - Upstart

- Originally created for Ubuntu, also used with RHEL6
- Builds upon SysV style, launches scripts based on events

Dependency-driven - Systemd

- Parallelizes as much as dependencies allow
- Unit files replace SysV init scripts
- Used in RHEL7, SLES12, Ubuntu 15.04

systemd System and Service Manager

Provides strict control of daemons and speedy booting

Natively uses "unit" files

- Compatible with SysV init scripts

Uses socket and D-Bus activation for starting daemons

- Aggressive parallelization with dependency support

Offers on-demand starting of daemons and daemon monitoring

- Captures all STDOUT and STDERR from daemons
- uses Linux `cgroups` to track daemon processes
- controls all enviromental runtime properties for daemons

Maintains mount and automount points

`systemctl` **- Administration command**

Used in RHEL7, SLES12, Debian 8, and Ubuntu since 15.04

- Many other distributions have adopted systemd (e.g. Arch)

systemd Targets

Targets allow grouping of units

- More flexible replacement for the SysV runlevel concept
- Defined in unit files ending in `.target`
- Use a *`unit_name`*`.wants/` directory to track grouped units

Runlevel targets provided for backwards compatibility

Special hardcoded targets provide core systemd functionality

Changing targets (runlevels)

- `systemctl isolate graphical.target`

Viewing & Changing the default boot target

- `systemctl get-default`
- `systemctl set-default` *`desired.target`*

Using systemd

Starting and Stopping a service:

- `systemctl start` *`unit_name`*`.service`
- `systemctl stop` *`unit_name`*`.service`
- `systemctl restart` *`unit_name`*`.service`

Enabling and Disabling a service:

- `systemctl enable` *`unit_name`*`.service`
- `systemctl disable` *`unit_name`*`.service`
- `systemctl mask` *`unit_name`*`.service`

Listing services and their state:

- `systemctl list-unit-files --type=service`

Modern `systemctl` **assumes the** `.service` **suffix if it is omitted**

Shutdown and Reboot

Activating `poweroff.target` **provides a graceful shutdown**

- **`systemctl poweroff`**
- **`shutdown`**
- Legacy commands: **`poweroff`**, **`halt`**, **`init 0`**

Activating `reboot.target` **provides a graceful reboot**

- **`systemctl reboot`**
- **`shutdown -r`**
- Legacy commands: **`reboot`**, **`init 6`**

Instant, non-graceful shutdown and reboot

- **`systemctl -ff halt`** or **`halt -f`**
- **`systemctl -ff reboot`** or **`reboot -f`**

System Messaging Commands

`write`

- Useful for sending short (1-2 line) instant messages to other users on the system
- Effective in a pipeline

`wall`

- Similar to **`write`**, but sends message to all users on the system
- Effective in a pipeline

`talk`

- Real-time keystroke at a time chat
- Works between Internet hosts as well

Controlling System Messaging

Terminal Devices

- Owned by special system group `tty`
- Have default group write permissions

The `mesg` Utility

- Toggles the terminal device's group write permission.
- Use **`mesg`** followed by **`y`** or **`n`** to toggle
- Use **`mesg`** with no arguments to see current status
- **`write`**, **`wall`** and **`talk`** commands honor current **`mesg`** status.

Lab 12

Estimated Time:

S12:	35 minutes
R7:	35 minutes
U1404:	35 minutes

Chapter

13

DETERMINE AND CONFIGURE HARDWARE SETTINGS

LPI Objectives Covered

101.1 Determine and configure hardware settings

Managing Linux Device Files

Usually in `/dev/`

Creating device files

- at install time (RPM)
- `mknod`
- dynamically with udev

Device file names

Hardware Discovery Tools

Manual discovery of hardware

- **dmesg**
- /var/log/dmesg
- /var/log/boot.msg
- /proc/ and /sys/
- **udevadm**
- **lspci**, **lscpu**, **lsscsi**, **lsusb**
- **dmidecode**, **biosdecode**
- **sensors**, **sensors-detect**

Configuring New Hardware with hwinfo

SUSE automatic detection: `hwinfo`
Runs at boot time and detects hardware changes
Automatically reconfigures system on addition or removal of hardware

- Uses detection routines found in the `/usr/lib64/libhd.so.*` libraries
- Records detected hardware in `/var/lib/hardware/`

PC Architecture and Bus

Processor and Architecture

- `/proc/cpuinfo`
- **`lscpu`**

Bus

- ISA
- PCI
- **`lspci`**, `/proc/pci`

DMA & IRQ

DMA

- `/proc/dma`
- **`hdparm -d`**

IRQ

- `/proc/interrupts`

USB Devices

`lsusb`

Kernel Messages

- **`dmesg`**
- RHEL7/SLES12: `/var/log/messages`
- U14.04: `/var/log/syslog`

USB Configuration

`udevd`

Kernel Modules

- Chipset Modules
- Device Modules

Disabling USB Storage

Configuring Kernel Components and Modules

Two methods of compiling kernel features

- Compiled into kernel binary installed under `/boot/`
- Separate kernel module (`*.ko`) installed under `/lib/modules/$(uname -r)`

Configuration options can be passed to kernel binary on boot

- Interactively from GRUB command prompt
- Persistently from GRUB config file

Configuration options can be passed to kernel modules

- Interactively when loading module
- Via files in the `/sys/modules/` directory
- Persistently from `/etc/modprobe.d/*.conf`

Kernel Modules

Kernel Modules Provide

- Hardware drivers
- Filesystems
- Network stack
- Linux Security Modules (LSM)

Discovering information about modules

- `modinfo`

Managing modules

- `lsmod`
- `insmod`
- `rmmod`

ABI compatibility

Handling Module Dependencies

Module Dependencies

- `/lib/modules/$(uname -r)/modules.dep`
- `depmod`

Inserting and Removing modules

- `modprobe`

Configuring the Kernel via /proc/

`/proc/`*`PID`*`/` **exposes information about each process on the system**

- Files and symlinks inside each folder provide information about the process

`/proc/sys/` **exposes tunable kernel parameters**

- view current values with **`cat`**
- modify with **`echo`**
- view and modify with **`sysctl`** command

Persistent tuning: `/etc/sysctl.conf`, `/etc/sysctl.d`

- system defaults:`/usr/lib/sysctl.d/*.conf`
- U1404 system defaults:`/usr/lib/sysctl.d/[123]0-*.conf`
- `systemd-sysctl.service`

Kernel Hardware Info – /sys/

Reasons for the creation of sysfs

- Provides hardware information needed by `udev`
- Centralized location for device information
- Clean up `/proc/`

sysfs

- Usually mounted on `/sys/`

`systool`

- List devices in sysfs by bus, class, and topology

```
systool -b scsi -v
systool -c scsi_host -v
systool -c scsi_disk -v
```

/sys/ Structure

Main sysfs directories

- `/sys/block/`
- `/sys/bus/`
- `/sys/class/`
- `/sys/devices/`
- `/sys/module/`

Other sysfs directories

- `/sys/firmware/`
- `/sys/kernel/`
- `/sys/power/`

Random Numbers and /dev/random

Random Numbers are very important

- Used as unique IDs in website cookies for session tracking
- Embedded in publicly accessible, yet "private" URLs
- Used in session keys during secure connection establishment

`/dev/random` **— collected pool of randomness (entropy)**

- Uses inter-interrupt timing as entropy source
  - Slow refill rate, especially on mostly idle systems
- Includes entropy from trusted hardware RNGs with kernel 3.16+
  - Only trusts VirtIO RNG by default with kernel 3.17+

`/dev/urandom` **— unlimited random data**

- Merely cryptographically strong, but acceptable for most applications

Lab 13

Estimated Time:

S12:	30 minutes
R7:	30 minutes
U1404:	30 minutes

This slide is intentionally blank.

Appendix A

LINUX FUNDAMENTALS

Unix and its Design Principles

Inherits features from Multics such as the hierarchical filesystem
Everything is a file
Small single-purpose programs
Ability to pipe small programs together to accomplish more complex tasks
The kernel makes minimum policy decisions, leaving things up to easily modifiable userland programs
All configuration data stored as text, (e.g. ASCII, UTF-8)

FSF and GNU

Richard Stallman – founder of GNU and the FSF

1983 – GNU (GNU's not Unix)

- goal: create the free GNU Operating System
- first programs: `emacs` and `gcc`

1985 – Free Software Foundation

- nonprofit organization for promotion of free software
- manages the GNU project

By 1991 the GNU system was almost complete

- only crucial component missing was a kernel

GPL – General Public License

Guarantees that free software remains free (as in freedom)

All software under the GPL makes source available to the end user

Changes to a GPL licensed software package must also be licensed under the GPL

Source code from GPL licensed software can be incorporated into other GPL licensed software

Other Licenses:

- http://www.gnu.org/licenses/license-list.html
- http://www.opensource.org/licenses/index.html

In 1992, Linus Torvalds released 0.12 of the Linux kernel under the GPL.

The Linux Kernel

Linus Torvalds – Finnish college student

- wanted to replace Minix, a UNIX-like feature-limited teaching OS

The Linux kernel

- fresh re-implementation of the UNIX APIs
- under the GPL license

The Linux kernel together with GNU and other programs forms a complete free operating system

Components of a Distribution

Typical Linux distributions provide

- collection of applications along with the Linux kernel
- installation program
- documentation
- support
- some are very specialized (e.g. Linux Router Project)
- POSIX and Single Unix Specification compliance

Most Linux distributions provide the same basic software:

- GNU software
  - GNU Coding Standards
- BSD and Linux utilities
- X.Org, GNOME, KDE, and other GUI components

Red Hat Linux Products

Invented the RPM Package Manager

Easy-to-use installer integrates partitioning and leverages RPM

Loyal to free software ideals: only ships open-source software with few exceptions that restrict modification and redistribution.

Fedora

- Cutting edge, community oriented project
- Provides new technology for future RHEL releases

Red Hat Enterprise Linux (RHEL)

- Enterprise targeted distribution with commercial support

CentOS

- Community edition of Red Hat Enterprise Linux

SUSE Linux Products

SUSE Linux Enterprise Family <http://www.suse.com>

- Server and Desktop releases
- 10-13 year maintenance life cycle
- Highly scalable, mature technology
- Three platforms: AMD64/Intel64, ppc64le (Power 8), IBM s/390x (z196 and z114)
- ISV certifications

The openSUSE Project <http://www.opensuse.org>

- Cutting edge
- 8 month release cycle
- Limited security updates for approximately 18 months

Debian

Second oldest active distribution
Initially sponsored by the FSF
Authored and Controlled by the Debian community
Very committed to free software
Uses own package management, dpkg
Innovated with in-place, no reboot upgradability
Easy to keep your system current

- `apt-get update`
- `apt-get upgrade`

Ubuntu

Founded by Mark Shuttleworth
Licensed by Canonical
Based closely on Debian
Uses Debian's package management, dpkg
Easy to keep your system current

- aptitude update
- aptitude upgrade

Package Update Availability

- Nine months
- LTS: Five years

Logging In

Serial terminals — Text mode login via serial port

- `mgetty`+`login` — Handles modems
- `agetty`+`login` — Handles VT100/VT220 dumb terminals

Virtual terminals — Text mode login(s) on local console

- `agetty`+`login`
- `mingetty`+`login`

Graphical — GUI login on local console

- `xdm`, `gdm`, `kdm`, etc.
- Terminal Emulator

  `xterm`, `rxvt`, `gnome-terminal`, `konsole`

Network logins — Remote text mode login

- `in.telnetd`+`login`, `in.rlogind`, `sshd`, etc.

got root?

Many operating systems have the concept of a super user

This super, or privileged, user has special access rights and privileges on the system

The `root` user is the privileged user on most Unix systems

Has the user ID (UID) of zero (`0`)

Switching User Contexts

`su`: launch a new shell as another user (using the target user's credentials)

- Use **`-`** | **`-l`** | **`--login`** to inherit login profile
- Default user is `root`

`sudo`: run a single command with another user's privilege

- Remembers authentication per-terminal (typically five minutes)
- Configuration affects authentication and available privilege (`/etc/sudoers`)

Gathering Login Session Info

Who are you really?

- UID – user id
- GID – group id
- terminal: tty, pts

Commands for gathering information:

- `id`

  ```
  id -un|whoami
  id -Gn|groups
  ```

- `tty`

Lab 1

Estimated Time:

S12:	15 minutes
R7:	15 minutes
U1404:	15 minutes