— LPIC General Linux Part 1 —

(Study Notes) [1] [2]

geoffrey hector robertson
geoffrey@zip.com.au

June 29, 2005

[1]Copyright ©2002 Geoffrey Robertson. Permission is granted to make and distribute verbatim copies or modified versions of this document provided that this copyright notice and this permission notice are preserved on all copies under the terms of the GNU General Public License as published by the Free Software Foundation—either version 2 of the License or (at your option) any later version.

[2]RCS Id = Id: lpic.general-linux-1.notes.tex,v 1.2 2002/02/28 23:06:04 geoffrey Exp geoffrey

Contents

Part I

Objectives

General Linux Part 1 [89]

The information in this section is taken from the LPI Project Objective Management System and is copyright to the Linux Professional Institute. It should be noted that the latest version of this information will be found at:
http://www.lpi.org/cgi-bin/poms.py.

A note about the numbering

- The numbers hard on the left margin with one dot, e.g. 1.103, are LPI Exam Topic numbers.
- The numbers in parentheses next to the LPI Exam Topic numbers are the old topic numbers and may be used to reference topics in older books and documentation.
- The numbers hard on the left margin with two dots, e.g. 1.103.1, are LPI Exam Objectives.
- The numbers in square brackets to the right of Topics and Objectives indicate weightings.

1.103 (1.3) GNU & Unix Commands [30]

1.103.1 Work on the command line [4]

Statement of Objective:

Candidate should be able to Interact with shells and commands using the command line. This includes typing valid commands and command sequences, defining, referencing and exporting environment variables, using command history and editing facilities, invoking commands in the path and outside the path, using command substitution, applying commands recursively through a directory tree and using man to find out about commands.

Key files, terms, and utilities include:

```
bash      echo  env   exec
export    man   pwd   set    unset
~/.bash_history       ~/.profile
```

Resources of interest

TBA

1.103.2 Process text streams using filters [7]

Statement of Objective:

Candidate should be able to apply filters to text streams. Tasks include sending text files and output streams through text utility filters to modify the output, and using standard UNIX commands found in the GNU textutils package.

Key files, terms, and utilities include:

```
cat     cut     expand   fmt      head
join    nl      od       paste    pr
sed     sort    split    tac      tail
tr      uniq    wc       unexpand
```

Resources of interest

TBA

1.103.3 Perform basic file management [2]

Statement of Objective:

Candidate should be able to use the basic UNIX commands to copy, move, and remove files and directories. Tasks include advanced file management operations such as copying multiple files recursively, removing directories recursively, and moving files that meet a wildcard pattern. This includes using simple and advanced wildcard specifications to refer to files, as well as using find to locate and act on files based on type, size, or time.

Key files, terms, and utilities include:

```
cp     find    mkdir    mv      ls
rm     rmdir  touch    file    globbing
```

Resources of Interest

Manipulating Files: Tutorial from LinuxCommand.org Find Manpage

1.103.4 Use streams, pipes, and redirects [3]

Statement of Objective:

Candidate should be able to redirect streams and connect them in order to efficiently process textual data. Tasks include redirecting standard input, standard output, and standard error, piping the output of one command to the input of another command, using the output of one command as arguments to another command and sending output to both stdout and a file.

Key files, terms, and utilities include:

```
tee   xargs   <    <<    >   >>   |  ‘ ‘
```

Resources of Interest

I/O Redirection: Tutorial from LinuxCommand.org
Tee Manpage

1.103.5 Create, monitor, and kill processes [7]

Statement of Objective:

Candidate should be able to manage processes. This includes knowing how to run jobs in the foreground and background, bring a job from the background to the foreground and vice versa, start a process that will run without being connected to a terminal and signal a program to continue running after logout. Tasks also include monitoring active processes, selecting and sorting processes for display, sending signals to processes, killing processes and identifying and killing X applications that did not terminate after the X session closed.

Key files, terms, and utilities include:

```
&    bg  fg  jobs  kill   nohup  ps  top
```

Resources of Interest

Job Control: Tutorial from LinuxCommand.org ps Manpage kill Manpage

1.103.6 Modify process execution priorities [2]

Statement of Objective:

Candidate should be able to manage process execution priorities. Tasks include running a program with higher or lower priority, determining the priority of a process and changing the priority of a running process.

Key files, terms, and utilities include:

```
nice   ps   renice   top
```

Resources of Interest

nice Manpage renice Manpage

1.103.7 Search text files using regular expressions [3]

Statement of Objective:

The candidate should be able to manipulate files and text data using regular expressions. This objective includes creating simple regular expressions containing several notational elements. It also includes using regular expression tools to perform searches through a filesystem or file content.

Key files, terms, and utilities include:

```
grep regexp sed
```

Resources of Interest

Regular Expressions for Poets: Courtesy of RobotWisdom.com A Tao of Regular Expressions: Courtesy of SiteScooper.org Capability Table: Showing a table of RegExp symbols available for Grep, Egrep, Ed, Vi, Sed, and Nawk.

1.103.8 Perform basic file editing operations using vi [2]

Statement of Objective:

Candidate must be able to edit text files using vi. This objective includes vi navigation, basic vi nodes, inserting, editing, deleting, copying, and finding text.

Key files, terms, and utilities include:

```
vi
/    ?
h    j    k    l
G    H    L
i    c    d    dd   p    o    a
ZZ   :w!  :q!  :e!
:!
```

Resources of Interest

VI Lovers Home Page: Resource listing for using the vi text editor

```
http://www.thomer.com/vi/vi.html
```

1.104 (2.4) Devices, Linux Filesystems, & FHS [21]

1.104.1 Create partitions and filesystems [3]

Statement of Objective:

Candidates should be able to configure disk partitions and then create filesystems on media such as hard disks. This objective includes using various mkfs commands to set up partitions to various filesystems, including ext2, ext3, reiserfs, vfat, and xfs.

Key files, terms, and utilities include:

```
fdisk   mkfs
```

Resources of Interest:

TBA

1.104.2 Maintain the integrity of filesystems [5]

Statement of Objective:

Candidates should be able to verify the integrity of filesystems, monitor free space and inodes, and repair simple filesystem problems. This objective includes the commands required to maintain a standard filesystem, as well as the extra data associated with a journaling filesystem.

Key files, terms, and utilities include:

```
du   df   fsck  e2fsck   mke2fs
debugfs   dumpe2fs   tune2fs
```

Resources of Interest:

TBA

1.104.3 Control mounting and unmounting filesystems [3]

Statement of Objective:

Candidates should be able to configure the mounting of a filesystem. This objective includes the ability to manually mount and unmount filesystems, configure filesystem mounting on bootup, and configure user mountable removable filesystems such as tape drives, floppies, and CDs.

Key files, terms, and utilities include:

```
/etc/fstab  mount  umount
```

Resources of Interest:

TBA

1.104.4 Managing disk quota [1]

Statement of Objective:

Candidates should be able to manage disk quotas for users. This objective includes setting up a disk quota for a filesystem, editing, checking, and generating user quota reports.

Key files, terms, and utilities include:

```
quota   edquota   repquota   quotaon
```

Resources of Interest:

http://www.linuxdoc.org/HOWTO/mini/Quota.html: The Quota mini-HOWTO

1.104.5 Use file permissions to control access to files [3]

Statement of Objective:

Candidates should be able to control file access through permissions. This objective includes access permissions on regular and special files as well as directories. Also included are access modes such as suid, sgid, and the sticky bit, the use of the group field to grant file access to workgroups, the immutable flag, and the default file creation mode.

Key files, terms, and utilities include:

`chmod` `umask` `chattr`

Resources of Interest:

TBA

1.104.6 Manage file ownership [2]

Statement of Objective:

Candidates should be able to control user and group ownership of files. This objective includes the ability to change the user and group owner of a file as well as the default group owner for new files.

Key files, terms, and utilities include:

`chmod` `chown` `chgrp`

Resources of Interest:

TBA

1.104.7 Create and change hard and symbolic links [2]

Statement of Objective:

Candidates should be able to create and manage hard and symbolic links to a file. This objective includes the ability to create and identify links, copy files through links, and use linked files to support system administration tasks.

Key files, terms, and utilities include:

`ln`

Resources of Interest:

TBA

1.104.8 Find system files & place files in the correct location [2]

Statement of Objective:

Candidates should be thoroughly familiar with the Filesystem Hierarchy Standard, including typical file locations and directory classifications. This objective includes the ability to find files and commands on a Linux system.

Key files, terms, and utilities include:

```
find   locate   slocate   updatedb
whereis   which   /etc/updatedb.conf
```

Resources of Interest:

TBA

1.106 (2.6)Boot, Initialisation, Shutdown and Runlevels [6]

1.106.1 (3) Boot the system

Statement of Objective:

Candidates should be able to guide the system through the booting process. This includes giving commands to the boot loader and giving options to the kernel at boot time, and checking the events in the log files.

Key files, terms, and utilities include:

```
dmesg  /var/log/messages
/etc/conf.modules or /etc/modules.conf
LILO  GRUB
```

Resources of Interest:

TBA

1.106.2 Change runlevels and shutdown or reboot system [3]

Statement of Objective:

Candidates should be able to manage the runlevel of the system. This objective includes changing to single user mode, shutdown or rebooting the system. Candidates should be able to alert users before switching runlevel, and properly terminate processes. This objective also includes setting the default runlevel.

Key files, terms, and utilities include:

`shutdown` `init` `/etc/inittab`

Resources of Interest:

TBA

1.108 (1.8) Documentation [8]

1.108.1 Use and manage local system documentation [5]

Statement of Objective:

Candidates should be able to use and administer the man facility and the material in `/usr/share/doc/`. This objective includes finding relevant man pages, searching man page sections, finding commands and man pages related to them, and configuring access to man sources and the man system. It also includes using system documentation stored in `/usr/share/doc/` and determining what documentation to keep in `/usr/share/doc/`.

Key files, terms, and utilities include:

`man` `apropos` `whatis` `MANPATH`

Resources of Interest:

TBA

1.108.2 Find Linux documentation on the Internet [2]

Statement of Objective:

Candidates should be able to find and use Linux documentation. This objective includes using Linux documentation at sources such as the Linux Documentation Project (LDP), vendor and third-party websites, newsgroups, newsgroup archives, and mailing lists.

Key files, terms, and utilities include:

not applicable

Resources of Interest:

TBA

1.108.3 Write System Documentation [1]

Statement of Objective:

Write documentation and maintain logs for local conventions, procedures, configuration and configuration changes, file locations, applications, and shell scripts.

Key files, terms, and utilities include:

not applicable

Resources of Interest:

TBA

Note

Difficult to test.

1.111 (2.11) Administrative Tasks [24]

1.111.1 Manage users and group accounts and related system files [7]

Statement of Objective:

Candidate should be able to add, remove, suspend and change user accounts.
Tasks include to add and remove groups, to change user/group info in passwd/group databases. The objective also includes creating special purpose and limited accounts.

Key files, terms, and utilities include:

```
chageg passwd groupadd groupdel groupmod grpconv grpunconv passwd
pwconv pwunconv useradd userdel usermod /etc/passwd /etc/shadow
/etc/group /etc/gshadow
```

Resources of Interest:

Chapter 9 - Managing User Accounts: The Linux System Administrators' Guide
Manpages for useradd usermod userdel groupadd groupmod groupdel useradd passwd chage

1.111.2 Tune the user environment and system environment variables [4]

Statement of Objective:

Candidate should be able to modify global and user profiles. This includes setting environment variables, maintaining skel directories for new user accounts and setting command search path with the proper directory.

Key files, terms, and utilities include:

```
env export set unset /etc/profile /etc/skel
```

Resources of Interest:

TBA

1.111.3 Configure and use system log files to meet administrative and security needs [3]

Statement of Objective:

Candidate should be able to configure system logs. This objective includes managing the type and level of information logged, manually scanning log files for notable activity, monitoring log files, arranging for automatic rotation and archiving of logs and tracking down problems noted in logs.

Key files, terms, and utilities include:

```
logrotate tail -f /etc/syslog.conf /var/log/*
```

Resources of Interest:

TBA

1.111.4 Automate system administration tasks by scheduling jobs to run in the future [4]

Statement of Objective:

Candidate should be able to use `cron` or `anacron` to run jobs at regular intervals and to use at to run jobs at a specific time. Task include managing cron and at jobs and configuring user access to cron and at services.

Key files, terms, and utilities include:

```
at atq crontab /etc/anacrontab /etc/at.deny /etc/at.allow
  /etc/crontab /etc/cron.allow /etc/cron.deny /var/spool/cron/*
```

Resources of Interest:

TBA

1.111.5 Maintain an effective data backup strategy [3]

Statement of Objective:

Candidate should be able to plan a backup strategy and backup filesystems automatically to various media. Tasks include dumping a raw device to a file or vice versa, performing partial and manual backups, verifying the integrity of backup files and partially or fully restoring backups.

Key files, terms, and utilities include:

```
cpio dd dump restore tar
```

Resources of Interest:

TBA

1.111.6 Maintain system time [3]

Statement of Objective:

Candidate should be able to properly maintain the system time and synchronise the clock over NTP. Tasks include setting the system date and time, setting the BIOS clock to the correct time in UTC, configuring the correct timezone for the system and configuring the system to correct clock drift to match NTP clock.

Key files, terms, and utilities include:

```
date hwclock ntpd ntpdate /usr/share/zoneinfo /etc/timezone
/etc/localtime /etc/ntp.conf /etc/ntp.drift
```

Resources of Interest:

TBA

Part II

Resources

Chapter 103

GNU & Unix Commands

Old number: (1.3)
Weight: [30]

Work on the command line [4]

Process text streams using filters [7]

Perform basic file management [2]

Use streams, pipes, and redirects [3]

Create, monitor, and kill processes [7]

Modify process execution priorities [2]

Search text files using regular expressions [3]

Perform basic file editing using vi [2]

103.1 Work on the command line [4]

103.2 Process text streams using filters [7]

103.2.1 Text Filter Exercise

First catch some text

Locate a section of text to practice filtering through various filters. For example save the last 12 lines of the GPL license in a temporary file. Edit the file with `vi` and add some tabs and some extra blank lines. Also duplicate a few lines. Add some carriage returns to the ends of a few lines (in vi do this in edit mode with a Cntl-v Cntl-m. They should show up as M̂s in vi).

```
$ locate gpl-lic
/usr/share/doc/HTML/en/common/gpl-license
...
$ tail -12 /usr/share/doc/HTML/en/common/gpl-license >  /tmp/some.txt
$ cd /tmp
```

The file `some.txt`

```
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  'Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989^M
  Ty Coon,       President        of       Vice^M


This General Public License does not permit incorporating your program into
This General Public License does not permit incorporating your program into
This General Public License does not permit incorporating your program into
This General Public License does not permit incorporating your program into



proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

`cat` the file

Have a look at the man page for `cat`.

- Plain `cat` the file:

  ```
  $ cat some.txt
  ```

- `cat` the file with all lines numbered:

  ```
  $ cat -n some.txt
  ```

- `cat` the file with non-blank lines numbered:

  ```
  $ cat -b some.txt
  ```

- Check to see if there are any non printing characters:

```
$ cat -v some.txt
...
  <signature of Ty Coon>, 1 April 1989^M
...
```

- Display any tabs:

```
$ cat -T some.txt
...
  Ty Coon, ^IPresident ^Iof ^IVice
...
```

- Do a `-vET` all at once:

```
$ cat -A some.txt
...
  <signature of Ty Coon>, 1 April 1989^M$
  Ty Coon, ^IPresident ^Iof ^IVice^M$
...
```

- Strip out surplus blank lines:

```
$ cat -s some.txt
```

`tac` the file

Have a look at the man page for `tac`.
Try it out.

Remove duplicate lines with `uniq`

Have a look at the man page for `uniq`.

- Plain `uniq`

```
$ uniq some.txt
```

- Show the repetition count:

```
$ uniq -c some.txt
```

- Print only the repeated lines:

```
$ uniq -dc some.txt
```

Print lines from the beginning of a file with `head`

Have a look at the man page for `head`.
Try it out on `gpl-license`.

Print lines from the end of a file with `tail`

Have a look at the man page for `tail`.

- Try it out on `gpl-license`.
- Try the follow option:

  ```
  $ tail -f /var/log/messages
  ```

  Create a message in another console to see it work.

Isolate fields with `cut`

Have a look at the man page for `cut`.

- Use cut to display only the gecos field and the shell field of the `passwd` file:

  ```
  $ cut -d ":" -f5,7 /etc/passwd
  ```

Format the text with `fmt`

Have a look at the man page for `fmt`.

```
$ fmt -w 50 some.txt

$ fmt -t -w 60 some.txt
```

merge lines of files using `paste`

Have a look at the man page for `paste`.
Create two files and merge them.

```
$ cat > first
one
two
three
four
^D

$ cat > second
this
that
these and
those

$ paste first second
```

103.3 Perform basic file management [2]

103.4 Use streams, pipes, and redirects [3]

103.4.1 Create, monitor, and kill processes [7]

103.4.2 Modify process execution priorities [2]

103.5 Search text files using regular expressions [3]

Fun with Regular Expressions by Adrian J. Chung

`http://thelinuxgurus.org/regexp.html`

103.6 Perform basic file editing using vi [2]

Chapter 104

Devices, Linux Filesystems & FHS

Old number: (2.4)
Weight: [21]

Create partitions and filesystems [3]

Maintain the integrity of filesystems [5]

Control mounting and unmounting filesystems [3]

Managing disk quota [1]

Use file permissions to control access to files [3]

Manage file ownership [2]

Create and change hard and symbolic links [2]

Find system files and place files in the correct location [2]

104.1 Create partitions and filesystems [3]

104.2 Maintain the integrity of filesystems [5]

104.3 Control mounting and unmounting filesystems [3]

104.4 Managing disk quota [1]

104.5 Use file permissions to control access to files [3]

104.5.1 Ken Caldwell's Summary: Use file permissions to control access to files

Linux is a multi user operating system and therefore needs to provide a system whereby the users can control access to their files.

All users are given a unique User IDentification number (UID) and are assigned to at least one group(of users). Each group is identified by a Group IDentification number (GID). Frequently users are assigned to a group containing only one member (themselves) as their primary group. The system administrator can add a user to other groups such as may be convenient for example ″sales″, ″engineering″, ″finance″ or ″management″

Not all ″users″ of the system are natural people others such as bin, daemon, lp, fetchmail and nobody also exist.

About file access permissions

Any file created by a user will be owned by that user and belong to the current group of that user. That is to say the file will be tagged with its creator's UID and GID. The file will also be tagged with its (default) permissions according to the umask of its creator.

The permissions pertaining to an ordinary file are the permission to Read the file, the permission to edit or delete the file (Write) and the permission to eXecute the file. These permissions are specified for: 1 The file's owner (ie the User who created the file) 2 Members of the Group associated with the file. (as determined by the GID) 3 all Others

Permissions are shown in the long format output of the ls command as a nine character string such as, for example, rwxr-x–x. The first three characters represent the permissions of the User who created the file. In this case permission to Read, Write and eXecute the file. Members of the Group have Read and eXecute permission while Others have only eXecute permission. Permissions can also be expressed as an octal number with one digit for the user, one for the group and one for the others. Read permission is given a value of 4, write permission a value of 2 and execute permission a value of 1. In our previous example the file could be described as having the permissions 751.

Permissions are interpreted slightly differently when applied to directories. The read permission is interpreted to mean the ability to list the directory. The write permission is interpreted to mean the ability to write files to, or delete files from, the directory. The execute permission allows a user to cd to that directory or to include it in a path to a directory to which you wish to change.

The initial permissions of a file upon creation are determined by subtracting the user's umask from 777. The default umask is usually 002 on systems where users have their own exclusive group or 022 otherwise. In the former case files will be created with rwxrwxr-x permissions and in the latter case rwxr-xr-x.

The permissions of a file may be altered by the file's owner by means of the chmod command. The chmod command is invoked as:

$ chmod (required change) filename

The bit in brackets can be either the octal value of the new permissions e.g. 644 or a string made up of three elements. The first element is one or more of u, g, o or a standing for User, Group, Others or All. The second element is +, - or = meaning add the designated access, remove the designated access or set exact access specified. The third element is the access type ie one or more of r,w or x.

There are three further access modes which we haven't discussed so far. They are SUID, SGID and the sticky bit. These are also expressed as an octal digit, 4 for SUID, 2 for SGID and 1 for the "sticky bit". Thus an executable file which we have previously described as having permissions 775 could more exactly be described as having permissions 0755. If the file was SUID the description would be 4755. Again the meanings are different for ordinary files and directories.

An executable file with the SUID bit set runs with the UID of the file owner instead of inheriting the UID of the parent process. Similarly for the SGID bit. The "sticky bit" has no meaning on Linux systems when applied to ordinary files. If the "sticky bit" is set on a directory then only the owner of a file may delete it from that directory even if the directory is world writable. This is most often seen on the /tmp directory. If the SGID bit is set on a directory then all files created in that directory will be assigned the GID of the directory rather than the GID of the user. Another way of creating files with a desired GID is to change the user's GID with the newgrp command.

104.6 Manage file ownership [2]

104.6.1 Ken Caldwell's Summary: Managing file ownership

The UID and/or the GID of a file may be changed by the file's owner by means of the chown command. man chown for all the options but typically invoked as:

$ chown newowner:newgroup filename

or:

$ chown 314:42 filename

chgrp is similar but may only be used to change the group.

104.7 Create and change hard and symbolic links [2]

104.8 Find system files and place files in the correct location [2]

104.8.1 Ken Foskey's Summary: Using `find`

:vi tw=72

...heading ... The find command.

The find command is one of the fundamental tools of Unix. It is a tool that is constantly rediscovered as you perform more and more complex operations with it. The man page of this simple tool is 555 lines long.

The most basic use of find is:

find ¡directory¿ -name "<mask>"

To find a missing file somewhere in you home directory

find -name missing.file

where is shorthand for your home directory. You can also use masks like "*.txt.gz" but you must put it in quotes.

Why do you have to put it in quotes?

...pause for discussion from floor...

When you use an * in a bash command line it is interpreted as a file expansion and it is looked for in the current directory and if it does exist it is substituted before the command is sent to find. If it is not found then your shell may generate an error message (for example csh, I think).

... page ...

According to the man page 'find - search for files in a directory hierarchy' This is true but you can also find directories as well, like the filesystems .

First we will start with some basic options:

Doing options: -print list the filename (default, never really use it). -exec run a command -ok run a command after prompting for confirmation. -ls list file like 'ls -dils', is a lot of file information.

Advanced doing options, I am sure you will use these one day:

-prune don't descend past this directory. -printf Print a filename based on format like C printf. -print0 print but end with a null character. -fprintf ¡fn¿ print a format string to a filename, (scripting??) -fprint ¡fn¿ print filenames to a file. -fls ¡fn¿ ls to a file

Options on what entries we select:

Most of these options take a number, +number, -number. A little explanation is required first.

picking one option, -atime:

-atime 2 Will pick any file accessed two days ago. -atime -2 Will pick any file access more than two days ago -atime +2 Will pick any file accessed in the last day.

** file date and time

Some basic stuff based on the file details itself.

-atime n files on access date -ctime n files on creation date (note chmod mucks this up) -mtime n files on modification date

-anewer n files on access date based on another file. -cnewer n files on creation date based on another file. -newer n files on modification date based on another file.

example: delete all files older than 7 days in the /data directory who have a .A extension.

Write the solution here.

A script may run a command and then 'touch' a tag file to give a timestamp when it was run. assume that the last thing a script does is touch modification.tag in the /parms directory. Write a command line that lists all details of files modified in the /apps/source/ directory based on this tag file.

.... pause to get suggestions from floor... .... lecturers note solution is find /data -mnewer /parms/modification.tag -ls....

Write solution here.

There is also a amin, cmin and mmin version of the above.

** Owner and group.

One problem with the Unix authentification system, when you delete a userid you end up with magic numbers on a directory listing. It is handy to be able to change the ownership on all files from the exiting staff member to the new person working on those projects.

-nouser users numeric id does not have and entry in /etc/passwd -nogroup group numeric id does not have an entry in /etc/group -uid n User by number -user name User by name -gid n Group by number -group name Group by name

I recently converted from Redhat to Debian. I installed a new harddisk and mounted the old one as /mnt/old1. I notice that when I do ls -al I get a username of 500 in the directory listing. Change all the occurrences of 500 to the username of ken.

.... lecturers note solution is AS ROOT

```
find /mnt/old1 -uid 500 -exec chown ken {} \;....
```

Write solution here

** Inode number and links

You have a directory listing, the hard link count is greater than 1. ... lecturers note wait and ask class how we know this ....

You have no idea where the other hard link is and you want to locate the other version to see what impact a change may have.

-inode n

... lecturers note, I have no idea how to do this ....

Write solution here

Advanced options on what entries we select:

-iregex Use regex rather than standard file masks.

Options on how we go through the directories: -xdev don't go into other file systems.

104.8.2 Andrew Eager's Summary: Using `locate`, `updatedb` and `slocate`

SLOCATE - Secure Locate

LOCATE - Normal Locate (Normally symlinked to slocate)

Slocate is used to find files on the system without actually having to search the entire directory tree. A database of all files on the system is created and is then used by slocate to reveal the files actual location. It is important to note that slocate may return a result which is no longer valid since the directory structure may have been modified since the slocate database was last created. For example, you create a file called poobar.txt, create the slocate database and then remove poobar.txt. Slocate will still return poobar.txt?s original location until the slocate database is recreated.

Slocate can be used in two modes:

- Search mode:- To locate an actual file within the database
- Database creation mode:- To build the database

Search usage:

```
slocate [-qi] [-d <path>] [-r <regexp>] <search string>...
```

`-q` Quiet mode. Suppress error messages.

`-i` Does a case insensitive search.

`-d` Specify a database to use.

`-r` Pass a regular expression instead of a search string.

Examples:

`locate ls`
```
$ locate ls ↩
...
 /etc/X11/xkb/symbols/xfree68/ataritt
/etc/X11/xkb/symbols/xfree68/amiga
/etc/alternatives/tclsh
...
```

`locate -r "/ls$"`
```
$ locate -r "/ls$" ↩
/home/geoffrey/tafe/mos/compress/ls
/usr/lib/bitchx/help/8_Scripts/ls
/bin/ls
```

The above example illustrates the need for a regex option to `locate`. In the first example there will be lots of hits. In the second there is only one (the actual `ls` command).

As well as searching for a file in the database, `locate` can also build the search database.

Database Creation Usage:

As well as searching for a file in the database, slocate can also build the search database.

-u Create slocate database starting at path `/`.

-U <**dir**> Create slocate database starting at path < *dir* >.

-c Parse original GNU Locate's `/etc/updatedb.conf`

-e <**dir1...**> Exclude directories from the slocate database when using the -u or -U options.

-f <**fs...**> Exclude file system types from the slocate database

-l Security level. 0–> security off, 1–> security on

-q Quiet mode. Error messages are suppressed.

-o <**file**> Specify the name of the database file to create

-v Be verbose

Examples

- Create a database for all directories under `/usr` and place the resulting database file into `slocate.db` in andy's home directory.

  ```
  # slocate -U /usr -o /home/andy/slocate.db  ←˒
  ```

- Create a database for all directories under /usr, excluding directories under /usr/man and place the resulting database file into slocate.db in andy?s home directory.

  ```
  # slocate  -U /usr   -e /usr/man  -o /home/andy/slocate.db ←˒
  ```

Update `slocate` database—`update`

`updatedb` is simply a link to slocate that implies the -u option. (Excerpt from the man page:- man updatedb)

```
$ ls -l ‘which updatedb‘ ←˒
lrwxrwxrwx  1 root  root 7 Mar 27 10:44 /usr/bin/updatedb -> slocate*
```

`updatedb` is typically executed periodically via cron.

`/etc/updatedb.conf`

- The `updatedb` (or `slocate`) tool can use a configuration file to decide which directories and file systems are included when the database is created. This file is normally located in `/etc/updatedb.conf`
- The following is a list of keywords that are recognised by updatedb (slocate) and their equivalent command line options

PRUNEFS <fs_type1 fs_type2...> - Option -f

PRUNEPATHS <dir1 dir2 dir3...> - Opion -e

- Example updatedb.conf

```
PRUNEFS="devpts NFS nfs afs proc smbfs autofs auto iso9660"
PRUNEPATHS="/tmp /usr/tmp /var/tmp /afs /net?
export PRUNEFS
export PRUNEPATHS
```

slocate Exercises

1. Create an slocate database in your home directory including all directories from / down.
2. Using the database created in step 1, locate all files with rm in the filename
3. Using the database created in step 1, locate the executable file rm using a regex. (ie /some/path/rm)
4. Create an slocate database in your home directory include all directories from / down but excluding the /bin directory.
5. Repeat (2) and (3) above. Do you notice anything different ?
6. After backing up your existing /etc/updatedb.conf, say

   ```
   # cp /etc/updatedb.conf /etc/updatedb.conf.orig ↩
   ```

   edit /etc/updatedb.conf to perform the same actions as in step (4).
7. When you have finished this exercise restore your original /etc/updatedb.conf.

Chapter 111

Administrative Tasks

111.5 Maintain an effective data backup strategy [3]

111.5.1 Backup Overview

Prepared by Grant Parnell

Decide what data is important and how long you can do without it.

- Is this used 24 x 7 or just business hours?
- During business hours how long can you do without it? 4 hours, 30 minutes, 5 minutes?
- How up-to-date is it required to get you running in an emergency?
- Are you backing up for archival or high availability or espionage?

Examples of Data

Static: Configurations of running servers. You need these 24x7 but they don't change much.

Databases / Transactions - financial & otherwise: These are updated frequently and need to balance. Associated with these are logs and duplication and other means of rollback and integrity checking. With databases it's often a good idea to dump them in a good portable format, especially if the inbuilt format is not cross platform or cross version compatible. EG 'mysqldump mydata ¿mydata.dump' will give you a text file which can be used on most mysql versions and possibly adapted to other database packages.

Logs: People don't tend to read them unless something goes wrong in which case they're valuable. These need to be kept but don't need to be restored in a hurry.

Home directories: This is a mixed bag of everything but some policies could be instated to make the admin's life easier. EG Making specific subdirectories for things and assigning them different backup/restore priorities. Often the existence of a home directory is more important than the rest of the contents as it may make a user unable to login without it.

Code repositories: Programmers should be accustomed to doing regular backups anyway, they often need to revert to an old version to figure out what they broke. Any tools used such as CVS that have a central repository should be backed up almost as often as programmers commit code, at least once a day but they could probably cope with it being missing for half a day.

High availability - read only: Websites frequently used by your clients. They can contain dynamic data but customers don't update it. This sort of scenario lends itself to frequent replication to a backup server.

High availability - interactive: Taking a website again, this one might allow the customer to do such things as place orders. The website maintains some state information to allow building of an order. This is the most difficult, the state information can be stored in a replicated database. In the event of web server failure the other one comes into play and the customer may have to login again but the information is kept. (Otherwise complex designs and expensive hardware can be used to seamlessly migrate the state to the other webserver).

Important Linux directories

```
/var/spool/mail   - daily backup
/var/lib/mysql    - databases - backup the dumps, and possibly
                    the binary.
/var/log ?        - from "don't care" to "backup daily"
/etc              - backup config changes
/home             - be selective, but if you can't backup daily.
/home/<user>/mail - contains the user's mail folders
                    (may also be 'Mail' or 'Maildir')
/home/<user>/.ssh - If you login using ssh keys only, this is a
                    must have.
/usr/local        - locally installed apps & data
                    Application specifics
```

111.5.2 Backup & Restore methods

Copy the files to another directory

This is the poor mans backup and does not offer much peace of mind. It does protect against accidental deletion and corruption by users. One advantage is that it can be very quick for things such as log files. You can also keep multiple copies, one for every day of the week for example. See /etc/logrotate.conf.

Backup to a standby partition

This has about the same level of peace of mind as the above. The backup partition can be left un-mounted after the backup. The backup is slower than the above but the restore operation can be quick. See also "Broken Mirror" method below.

Backup to tape

This is probably the most common backup used in the commercial world. It's easy to backup the lot every day provided you have the tape capacity. If you don't, you become more selective as to what to backup. There's a variety of software to do this but there's 3 main basic systems. Tar, cpio and dump. Often commercial software uses these basic systems and provide for labelling and indexing as well as multi-server capability from a simple GUI. The reason for using the basic systems is you can restore from them if you have to.

Backup to standby disk

This can offer peace of mind and a fairly cheap backup for people that don't require 24x7 service. Basically a removable drive bay houses another hard disk of similar capacity and the entire system is backed up. This can be done partition by partition or file by file using dd, cpio or rsync. Additional steps can be taken to ensure that the backup is also bootable. The backup drive should be removed once done and treated like a tape. The disadvantage here is that you most likely will need to power down the system twice for one backup. Alternately, if you have an external USB or fire-wire storage medium it becomes possible to do this without downtime.

Backup to CDROM/DVD

Under Linux (as far as I know) there's no software to directly write data without creating an image first. This means there must be sufficient space available. It would be possible to create a bootable CD with restore software and a compressed filesystem but I haven't seen this. It may be OK if you don't have a large filesystem or you have a DVD writer or you're not backing up everything.

RAID System

Not strictly a backup but a RAID system can protect against hard drive failure by providing redundancy. Data is written simultaneously to 2 or more hard drives and can include parity information. It does not protect against corrupt databases and people removing files. It will corrupt and remove files equally well on all disks. Linux can do RAID in software very well but the ideal is a hardware solution involving hot swapable disks so they can be replaced while the system is fully running. A RAID system can mean the difference between going on-site at 3am and saying "Oh dear, we'll replace that first thing in the morning". Just ensure that you do have a replacement readily available and do not have to wait a week.

RAID Tape array

In a similar manner to RAID 5 disks, data is written in parallel to 5 tape drives which increases throughput and data integrity.

Backup Server

All of the methods discussed so far involve direct transfer from server to backup medium. If you have a number of servers it may not be practical to install backup devices on each. Another way is to remotely access the required medium directly (/dev/rmt0) but arbitration of access can be an issue. An increasingly popular way is to provide a super-server with a huge amount of disk space capable of holding everything required by the other servers. Transferring the data can happen at any time in either a batch or continuous process. A batch would be say backup a whole directory at once whereas a continuous operation might be transmitting log information or database updates. The backup server itself may then employ any one or more methods to perform backups of itself, possibly based on some statistical analysis. An example of this is a system called ADSM which employs RAID arrays, multiple tape drives, a tape robot with barcode reader and intelligent software that tells the operators which tapes are to go off-site and which ones it wants back. It essentially is a huge cache that stores frequently changing data locally and stores old data off-site.

Broken Mirror

If you've got about 100Gb of data on a mirrored pair of disks and only have a 10 minute backup window this may be for you. Basically you bring the system down, unhook one of the mirrors and replace it with another set of drives and bring the system up again. Mirroring starts from scratch during quiet time and should be finished before load picks up again. With the drive set you just un-hooked this can then be loaded into the standby server and backed up to tape over the course of many hours. Some high end servers can perform this operation without downtime as the hooking up can be done using inbuilt hardware or such things as dual-port fire-wire drive bays. All that is required in this case is an application shutdown, sync, dismount, remount, application start type operation.

111.5.3 Software

dd - can be used to copy raw disk blocks, even to tape (yuk). eg dd if=/dev/hda1 of=/dev/hdb1

tar - Tape ARchive - you all know how to unpack tgz files, and maybe even create them. Just remove the 'f' option. It also can be an advantage not to use compression as some drives have this built in. Also, a portion of the tape being corrupt can ruin the rest of the data, whereas you can skip corrupt bits and pickup the next file if not compressed. eg tar -c /home cd /tmp; tar -x

cpio - cp I/O - Similar capabilities of tar but different methodology. EG find /home — cpio -oB ¿/dev/tape cd /tmp; cpio -idB ¡/dev/tape

rsync - remote sync - can sync a directory or whole filesystem by only transferring the changes between them. Be careful about trailing slashes. rsync -a /home /backup/ rsync -a -e ssh /home backup@backup:/serverA/

Arkeia - commercial package BRU - commercial package Amanda - Open source? Thousands more, some are client/server model and can backup multiple operating systems which is great.

See http://www.linuxhelp.com.au/free.shtml for our generic CPIO backup script.

111.5.4 Rotation & off-site strategies

It's no good having a backup if it's sitting next to the computer when there's a fire. You've got to have some off-site backups for really important stuff. On a small scale a friend of mine has a backup of all my music CD's I couldn't live without.

You could use this example strategy with any bulk medium but typically people refer to tapes or a set of tapes and for convenience I'll refer to a tape. If you can fit everything on one tape good for you, life is easy, backup the lot daily. If you don't you'll have to do an incremental backup (ie what's changed) daily and do a whole backup with multiple tapes weekly. Take the weekly backup off-site home from work or over to a trustworthy friend's place. Once a month take a weekly backup to long term storage and keep it for 7 years or something if it's got all your tax info on it. It goes without saying the tapes should be labelled full/incremental and a date, hostname and what sequence in the set they are. Daily backup tapes may be rotated once a week with a new tape supplied once a week for a specific day of the week. Eg week1 will be all new tapes with one shipped off on Monday morning. week2 it'll be a new tape for Sunday morning, week3 it'll be Saturday mornings tape that's new. Alternately, some people believe the weekly or monthly should be on a fresh tape that's never been used.

With this strategy you get reasonable rotation of the tapes keeping costs down and for archival purposes, if you keep at least a months worth of data on the server you'll be able to go back to any point over the last few years and pull out a file. If you keep at least 3 months on hard disk you'll have 3 copies of this on 3 separate tapes because believe it or not they do fail and it will happen to you. To explain this more fully lets look at the following table and assume we have some wages data every week and the company's just started and there's 4 weeks per month.

	server has	weekly tape has	monthly tape has
wk1	wk1	wk1	-
wk2	wk1-2	wk1-2	-
wk3	wk1-3	wk1-3	-
wk4	wk1-4	wk1-4,month1	wk1-4,month1
wk5	wk1-5	wk1-5,month1	-
wk6	wk1-6	wk1-6,month1	-
wk7	wk1-7	wk1-7,month1	-
wk8	wk1-8	wk1-8,month1-2	wk1-8,month1-2
wk9	wk1-9	wk1-9,month1-2	-
wk10	wk1-10	wk1-10,month1-2	-
wk11	wk1-11	wk1-11,month1-2	-
wk12	wk1-12	wk1-12,month1-3	wk1-12,month1-3
wk13	wk2-13	wk2-13,month1-3	-
wk14	wk3-14	wk3-14,month1-3	-
wk15	wk4-15	wk4-15,month1-3	-
wk16	wk5-16	wk5-16,month2-4	wk5-16,month2-4
wk17	wk6-17	wk6-17,month2-4	-
wk18	wk7-18	wk7-18,month2-4	-
wk19	wk8-19	wk8-19,month2-4	-
wk20	wk9-20	wk9-20,month3-5	wk9-20,month3-5
wk21	wk10-21	wk10-21,month3-5	-
....			

A complete backup and archive strategy should provide a means of going back to any point in time for critical data. Sometimes keeping the whole lot of data is not required. For example you could drop the weekly information and keep the monthly summary information and do a dedicated monthly backup for this data. The monthly data may be optimised and arranged for searching and an index provided but essentially contain all the information from the weekly data.

Part III

Practical Exercises

Chapter 103

GNU & Unix Commands

Old number: (1.3)
Weight: [30]

Work on the command line [4]

Process text streams using filters [7]

Perform basic file management [2]

Use streams, pipes, and redirects [3]

Create, monitor, and kill processes [7]

Modify process execution priorities [2]

Search text files using regular expressions [3]

Perform basic file editing using vi [2]

103.1 Work on the command line [4]

103.2 Process text streams using filters [7]

103.3 Perform basic file management [2]

103.4 Use streams, pipes, and redirects [3]

103.5 Create, monitor, and kill processes [7]

103.6 Modify process execution priorities [2]

1. This exercise requires a few example processes to play around with, so first we will create a few.

   - Using the `vi` editor (for practice) make a file called `a.c`.
     - Do not use the arrow keys—use h, j, k and l.
     - Do not use the Backspace or Delete keys—use `x`.
     - Be sure you are familiar with the use of `i`, `a`, `o` and `O` for entering new text.
     - For saving and quitting use each of `ZZ`, `:w`, `q:` and `!`.
     - Delete, Yank and Put with `dd`, `yy` and `p`.
     - Use `u` to undo, numerical modifiers (e.g. 5dd) and `.` to repeat.
   - The file should contain this text:

     ```
     #include <stdio.h>

     int main()
     {
       int i = 0;

       while (1) {
         system("clear");
         printf("Process a: %d\n", i);
         ++i;
       }

       return 0;
     }
     ```

   - Copy the file `a.c` to `b.c` and `c.c`.

     ```
     $ cp a.c b,c.c ↩
     ```

   - Edit the `printf()` calls in files `b.c` and `c.c` to refer to `Process b:` and `Process c:` respectively.
   - Compile the programs:

     ```
     $ gcc a.c -o a; gcc b.c -o b; gcc c.c -o c ↩
     ```

2. Test your three programs.

   - Start a program running in the background:

     ```
     $ ./a ↩
      Process a: 400
     ```

   - Suspend the program:

     ```
     ^z
     [1]+  Stopped                 ./a
     ```

   - Kill the job:

```
$ kill %1 ←↩
```

3. Use `ps` to view the processes.

   - Run the processes in the background with differing degrees of niceness:

     ```
     $ ./a& nice ./b& nice -19 ./c& ←↩
     ```

   - Glance at and absorb the manpage for `ps`. Note well that some options are preceded by a minus(-) and others are not. For example `ps a` and `ps -a` do different things.
   - Locate the processes using the `ps` command; try the `a`, `u`, `x`, and `f` options separately and in combinations. e.g.:

     ```
     $ ps aux |grep "./a " ←↩
     ```

   - Kill and restart one of the processes (use the PID not "123456":

     ```
     $ kill 123456  ←↩
     $ ps aux |grep "./a " ←↩
     $ ./a&
     $ ps aux |grep "./a " ←↩
     ```

4. Use `top` to view and modify the processes.

   - Run `top` in the foreground:

     ```
     $ top ←↩
     ```

   - Look at the `top` help: Press h
   - Sort by accumulated time: Press `T`
   - Re-nice a process (Note: Users may only monotonically increase the niceness processes, and (&&) they must own the process.):
     - Press r
     - PID to renice: 1234567890 ←↩
     - Renice PID 1973 to value: 5 ←↩

5. Reniceing from the command line:

   - After finding it's PID`renice` one of your processes:

     ```
     $ ps aux |grep "./c " ←↩
     $ renice +15 1234567890
     ```

   - Re-nice negatively—notice that only the superuser may reduce the niceness of a process.

     ```
     $ renice -10 1234567890 ←↩
     renice: 1234567890: setpriority: Permission denied
     $ su -c 'renice -10 1234567890' ←↩
     ```

6. Kill off `./a`, `./b` and `./c`.

103.7 Search text files using regular expressions [3]

103.8 Perform basic file editing using vi [2]

103.8.1 Introduction to `vi`

Note

Should you need a good sized text file to practice editing on, you will almost certainly find a copy of the GPL License on your system. Be sure to deconsecrate the file by renaming before munging it. You may locate one thus:

```
$ locate GPL ↩
...
/usr/share/doc/netpbm-9.14/GPL_LICENSE.txt
/usr/share/doc/cdda2wav-1.10/GPL
/usr/share/doc/cdparanoia-alpha9.8/GPL
/usr/share/doc/stunnel-3.19/COPYRIGHT.GPL
/usr/share/doc/libesmtp-0.8.4/COPYING.GPL
...
```

Make a scratch copy at a suitable location and open it for editing:

```
$ cp /usr/share/doc/stunnel-3.19/COPYRIGHT.GPL  ↩
/tmp/munged_gpl.txt ↩
$ vi /tmp/munged_gpl.txt ↩
```

103.8.2 Vi tour

vi

Vi is the Unix editor, simply it is available on just about every Unix installation by default. This is the reason that you have to have a minimum of basic vi skills to allow you to change files when you favourite editor is not available, boot disks have vi due to lack of space.

Please note that there are many implementations of vi, the baseline is a very crude editor. This extends to a very extended and powerful VIM editor available on most platforms. I use vim because I am restricted to vi on many of the Unix systems I support, rather than trying to 'switch editors' mentally I have vi on every platform I use, including Windows.

vi is a mode editor

Vi is an editor and it is definitely not a friendly editing environment. In fact a random typing of keys of the keyboard can render your text totally unreadable. Vi has been around a long time it was created in an era where editors were modal. The editor can be in three different unrelated states described below.

Input mode This is the mode where you simply type. You characters appear in the text file as you type.

Command mode This is where character take on special significance, for example 'i' for insert 'D' to delete to the end of the line.

Line mode This is the mode when you press a : in command mode, this is where you can type some powerful (and sed like) commands to alter the document.

Rather than bore you with yet another description I will give you a series of exercises to work through showing each command. In the examples ¡esc¿ means to press the escape key. This will return you to command mode or cancel an action in other modes.

Inserting text

There are at least three ways to insert text into a document in vi, the following exercises will take you through the basic commands.

Exercise 1 - insert Create a text document, just to get you used to switching in and out of the insert and command mode.

```
$ vi test.txt ↩
ithis is the Linux course<return>
We want a few lines of txt to work with. <esc>
```

The i inserts text before the cursor.

Exercise 2 - open We want to add some extra text to the document, we want to enter it on the line after the current line, we use the open command by pressing `o`.

```
oI am adding an line after the current line.<esc>
```

Exercise 3 - append Now I want ot continue adding another sentence on the current line. If I press 'i' I will insert before the full stop, in this case I want to append it after the full stop.

```
a I want to add another sentence.<esc>
```

You line should now look like:

```
I am adding an line after the current line. I want to
add another sentence.
```

Movement keys & multiplication

There are a huge number of ways to move around in vi, the arrow keys will not always work . You should be aware of the single character work arounds for these in case your terminal is not set up properly on the box you are 'telnet'ing or 'ssh'ing from or to.

h Left one character *

j Down one line *

k Up one line *

l Left one character *

w Forward one word

b Back one word

e End of current word

G End of file *

nG Goto a particular line *

{ Back one paragraph

} Forward one paragraph

$ End of current line

ˆ First non blank character in line

0 Beginning of line (also `|`)

Ones marked with an asterisk are required by POMS, Personally I find word movement is my main tool here. I also use the control keys to scroll screens a far bit. These are listed on the printed cheat sheet.

ALL these movement commands can be prefixed by a number to multiply the effect of the movement. To move up 99 lines enter 99k.

Note that some vi editor have a status on the bottom line, sometime it will show that you have entered a multiplier, sometimes it is quiet but the results are _unexpected_.

Exercise 4 - back In the previous exercise I have forgotten a word after the an I wanted the word 'extra'. I have to go back that position in the line.

`12b` will get me back to the beginning of the word I want to insert before, this is an example of a count followed by a command.

Warning on the counts it can repeat most commands whether it makes sense or not at the time. For example enter the following:

```
4iextra <esc>
```

What happened? Why?

This is handy with asterisks `70i*<esc>`, try it.

`12b` moves the cursor back 12 words. A word is a sequence of characters separated by a space character like a ¡space¿ or a ¡tab¿ Also the punctuation characters or '.', '?', ';' and so on.

Exercise 5 - general movement Edit a large text file on your system. You should be able to do the following with two keystrokes.

- Jump to line 6
- Move 8 character to the right.
- Go to the end of the line

- Go to the beginning of the line.
- Jump to the top of the paragraph in the document (a paragraph is delimited by blank lines.)

You should be able to do this with one keystroke

- Move to the left margin To the end of the line
- Move to the end of the file.

The undo command

The real vi command undo is very very limited. It does not allow for a lot of recovery. There are two undo commands 'U' and 'u'.

The lower case u will undo the last action that you have done. This includes itself so it will cycle through doing and undoing.

The upper case U will undo the any change to the current line and restore it back to its original state.

Extended vi editors such as vim will allow multiple lower case u undo commands but do not rely on it on an unknown box. To get out of a problem you simply have to quit without with ':q!'.

No exercise here, I figure you will find this out without any help. Try both commands. Check the difference.

Deleting changing and copying text

d Delete *

c Change *

x Delete one character

y Yank (copy text)

Ones marked with an asterisk are required by POMS, Personally I use single character deletion all the time. I use line deletion all the time.

There delete command and yank command can be used with a movement to delete all text within this movement. For example d$ will delete to the end of the current line, y$ will yank it into a buffer. Delete will also store the deletion in a buffer to get it back quickly just use paste describe later.

To delete or yank a line simply repeat the d or y. To delete a single line use dd, to delete the current line and the next two use 3dd. To yank the next 10 lines use 10yy.

Exercise 6

Please your cursor in the middle of a line. Delete from the current position to the beginning of the line with two keystrokes.

Delete from the current position to the end of the line.

Note that there is a shorthand for d$ (delete, move to end of line) by using a capital D. Move to the centre of a line of text and try it.

Chapter 104

Devices, Linux Filesystems & FHS

Old number: (2.4)
Weight: [21]

Create partitions and filesystems [3]

Maintain the integrity of filesystems [5]

Control mounting and unmounting filesystems [3]

Managing disk quota [1]

Use file permissions to control access to files [3]

Manage file ownership [2]

Create and change hard and symbolic links [2]

Find system files and place files in the correct location [2]

104.1 Create partitions and filesystems [3]

104.1.1 Using `fdisk`

Care must be taken using `fdisk` as any changes to your disk's partition table will make existing data on the disk unaccessible. There is debate about what the "`f`" in `fdisk` stands for.

Using `fdisk` non-destructively on a hard disk

- View the partition table for the hard disk:

```
# fdisk -l /dev/hda ↩

Disk /dev/hda: 255 heads, 63 sectors, 3648 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot Start   End    Blocks   Id  System
/dev/hda1   *      1   768   6168928+   c  Win95 FAT32 (LBA)
/dev/hda2        769  3648  23133600    5  Extended
/dev/hda5        769   780     96358+  83  Linux
```

- Print the size of a partition in blocks (30G disk):

```
# fdisk -s /dev/hda ↩
29302560
# fdisk -s /dev/hda5 ↩
96358
```

Using `fdisk` on a diskette

Warning: it makes no sense to use `fdisk` on a floppy—this is just an exercise.

- Format the floppy disk:

```
# fdformat /dev/fd0 ↩
```

- Start fdisk using the floppy diskette:

```
# fdisk /dev/hda ↩
```

- Look at the menu:

```
Command (m for help): m ↩
```

- Print the partition table:

```
Command (m for help): p ↩

Disk /dev/fd0: 2 heads, 18 sectors, 80 cylinders
Units = cylinders of 36 * 512 bytes

    Device Boot    Start       End    Blocks   Id  System
```

- Use n for new and construct this partition table:

  ```
      Device Boot    Start       End    Blocks   Id  System
  /dev/fd0p1 *           1        20       351    1  FAT12
  /dev/fd0p2            21        25        90   83  Linux
  /dev/fd0p3            26        80       990    5  Extended
  /dev/fd0p5            26        30        81   83  Linux
  /dev/fd0p6            31        70       711   83  Linux
  /dev/fd0p7            71        80       171   83  Linux
  ```

- List the partition types:

  ```
  Command (m for help): l ↩
  ```

- Change the first partition to type 1:

  ```
  Command (m for help): t ↩
  ```

- Toggle the bootable flag on the first partition:

  ```
  Command (m for help): a ↩
  ```

Using `cfdisk`

`cfdisk` is reputably more user friendly than `fdisk`. There are some partition adjustments that may require `cfdisk`.

Have a look at your floppy diskette partitions:

```
# cfdisk /dev/fd0 ↩
```

Using `sfdisk`

`sfdisk` is interactive partition editor.

Have a look at your floppy diskette partitions:

```
# sfdisk /dev/fd0 ↩
```

You may wish to Cntl-C out of this program if you wish not to edit the partition table.

Using `GNU parted`

`parted` is a partition editor that can resize partitions.

- Have a look at your floppy diskette partitions:

  ```
  # parted /dev/fd0 ↩
  ```

- Check the menu:

  ```
  (parted) p ↩
  ```

- Experiment on you partitioned floppy.

104.2 Maintain the integrity of filesystems [5]

104.3 Control mounting and unmounting filesystems [3]

104.4 Managing disk quota [1]

104.5 Use file permissions to control access to files [3]

104.6 Manage file ownership [2]

104.7 Create and change hard and symbolic links [2]

104.8 Find system files and place files in the correct location [2]

Part IV

Questions

104.103 (1.3) GNU & Unix Commands [30]

104.103.1 Work on the command line [4]

104.103.2 Process text streams using filters [7]

104.103.3 Perform basic file management [2]

104.103.4 Use streams, pipes, and redirects [3]

104.103.5 Create, monitor, and kill processes [7]

104.103.6 Modify process execution priorities [2]

104.103.7 Search text files using regular expressions [3]

104.103.8 Perform basic file editing using vi [2]

1. How do you quit vi without saving? ______________________
2. In vi command mode what does the `p` key do? _____________
3. The vi editor has three modes. Name them:

   _________________ ___________________ ________________________

4. In vi insert mode what does the following key sequence do?
   `<esc>:wq↩` ____
5. How do you search forward for the next occurrence of the word "from" in vi. __________
6. What command will delete the next 23 lines of text in vi. __________
7. Which command inserts text into vi at the current cursor location?
   (a) `a`
   (b) `f`
   (c) `T`
   (d) `i`
8. Which way does cursor move when you press each of these keys in vi command mode?

   k ________________
   h ________________
   l ________________
   j ________________

9. In vi command mode what does the `u` key do?

   ____

10. In the vi editor what does the term "yank" mean.?

    ____

11. What do you type to repeat the last command? __________

12. In vi command mode to move a line from its current position to three lines down the page you would enter:

    (a) `3dd3lp`

    (b) `ddk3p`

    (c) `1dd3p`

    (d) `dd3kp`

13. This command moves three lines of text into a vi buffer named `a`:

    (a) `"a3dd`

    (b) `a3dd`

    (c) `"add3`

    (d) `A5dd`

14. The G command does what? ______________________________________

Part V

Meta

Chapter 105

Making Slides Using LaTeX

105.1 Making a

- Install the acutex package for emacs
- Open a new document:

```
$ emacs my.slides.tex ↩
```

- In emacs create a document environment: C-C C-E Enter for the default (document) in the mini buffer.
- For document style enter: `seminar`
- For options enter: `a4`

blah

- aaaaaaa
- dddddddd
- wwww
- cc

- aaaaaaa
- dddddddd
- wwww
- cc

blah de blah de blah