Support de Cours

Institut de la
filière numérique

Formation T2SI :

Technicien Supérieur de Support en Informatique

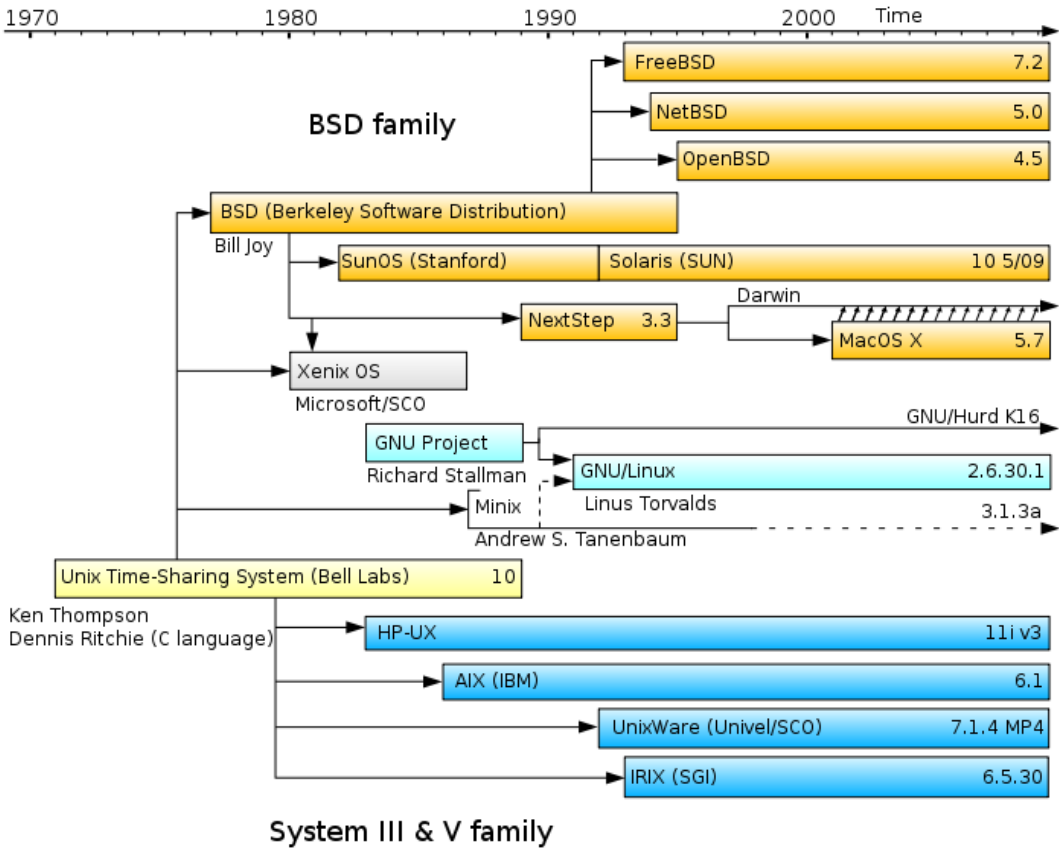
LINUX : DEBIAN (management)

Table of Contents

1 Introduction

1.1 History of Unix based systems

In 1969, the AT&T's Bell Laboratories in the United States conceived the Unix operating system. Unix led to the creation of Minix (an minimal Unix-like operating system, designed for education in computer science) and BSD (Berkeley Software Distribution), of which Mac OS X is derived.



In 1991 a finnish student called Linus Torvalds, frustrated by the licensing of MINIX limiting it to educational use only and preventing any commercial use, began to work on his own operating system, which eventually became the Linux kernel.

1.2 Definitions

Open source : the source code as well modified code must be made available (it does not mean that the distribution is free of charge, for example Redhat).

GNU : stands for Gnu is Not Unix, a free software, mass collaboration project, announced on September 27, 1983, by Richard Stallman at MIT. Early Linux kernel developers ported GNU code.

Flavours : implementations of Unix designed to work with different types of hardware, and having its own unique commands or features.

Distributions ("distros") : particular implementation, of the Linux operating system, packaged with a set of applications and a user interface

Kernel : central OS component and the bridge between a software application and its data

Desktop environment : unified graphical interface for an operating system (you actually see when using a computer : how windows are drawn and managed, panels/taskbars, menus, file management, and so forth)

1.3 Characteristics and distributions

Below are some key characteristics of Linux, which contributed to its success:

- cross-platform
- multitask
- multiuser
- open source kernel and components
- many specialised distributions driven by developpers and user communities
- stability (modular by design, not monolithic)
- security (separation of administrative privileges from normal user privileges)
- file system based on a unified scheme (all directories are attached to the root directory)

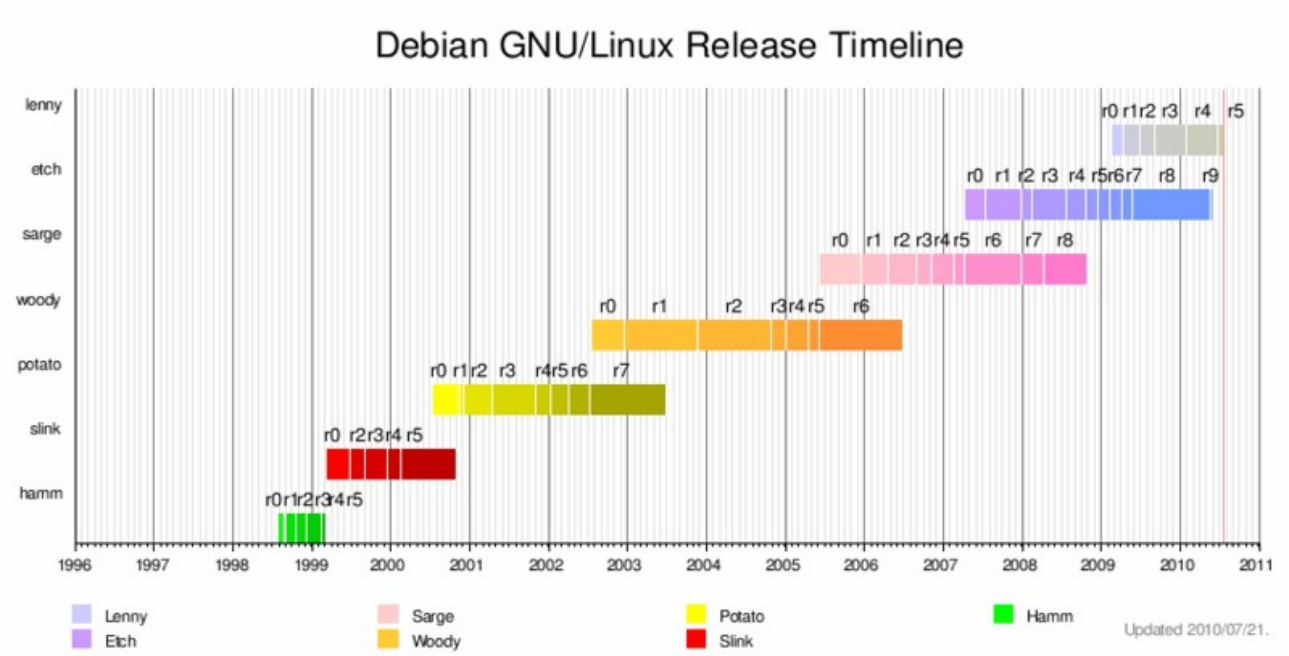
There are currently over six hundred Linux distributions. All are derivatives from 3 base distributions :

- Red Hat
- Slackware
- Debian

Some are commercially-backed distributions, such as Fedora (Red Hat), openSUSE (Novell), Ubuntu (Canonical Ltd.), and Mandriva Linux (Mandriva), and others are entirely community-driven distributions, such as Debian and Gentoo. Distributions have small variations in file structure and also differ by the type of packets (Debian packets are *.deb, Slackware packets need to be compiled, RedHat packets are *.rpm)

See : http://en.wikipedia.org/wiki/Linux_distribution

Debian is often considered one of the most stable version of Linux : indeed testing packets between each release takes more time than other distributions (2-3 years).



Note that all Debian versions are named after Toy Story characters.

2 Installation of Debian

2.1 Options

We will choose the following options to install Debian :

- graphic install
- leave domain empty
- provide credentials
- choose entire disk (not LVM : logical volume management, flexible volumes)
- select Guided Partitioning (separate /home /usr /var /tmp)

These volumes will be located on separate partitions, and /home /usr etc... are actually mount points. Motivation in partitionning is to separate logs (so they don't fill the hard disk) and the users home folder so that the system can be reinstalled without affecting user files.

- keep default size calculated by the installer

Note that the installer creates 1 primary partitions (number 1) and the others are logical partitions (numbering starts at 5 since there could be 3 more primary partitions). B means boot partitions. The default file system is ext3.

- packet manager: choose France and ftp.fr.debian.org (allowed by proxy)
- leave proxy blank

2.2 Navigation

Login with the normal user created during installation: the prompt gives some information about the user and the current folder :

```
username@computername:~$
username@computername:/home$
```

~ : represents user home folder (current location)
$: simple user
: root

Switch user using the command `su [username]`. Note that if you are logged in as a normal user, it will switch by default to root if no username is provided. Type `exit` to disconnects session (or return to the previously open session if you have used the command `su`.

`echo $HOME` : outputs path of user's home folder

All users have their folder in /home except root (/root)

Moving between folders with the command `cd`

```
cd ../..        goes up two levels
cd -            back to previous folder
cd ~            go to user home folder
cd /home        absolute path
cd foldername   relative path
```

Listing directory content with the command `ls` (a: hidden files, l:long format)

```
ls -al /etc
ls -al /etc | more
ls -al /etc | less        (can scroll up and down; q to exit)
ls -l /etc > file1   (outputs result of the cmd into the file)
```

To remove the beeps : `#rmmod pcspkr`

Command recall can be used with arrows and tab provides completion (or choice if ambiguous). Linux is a multitask system : it is possible to work in separate virtual consoles by pressing ALT + F1 etc..

2.3 *Installation of the desktop environment*

Switch to a separate virtual console and login as root. Then type the command :

```
#apt-get install gnome
```

Note that there are several desktop environments amongst which GNOME and KDE are the dominant solutions. The desktop environment is meant for the normal user. The root superuser is not allowed to open a session with the graphic interface for security reasons. Start the graphic interface by typing : `startx`

Install VMware tools (first install gcc and linux-headers), then either select install Vmware Tools and

extract archive from the CD or scp the archive from the network, then launch the script by typing : ./<scriptname>). See 13.1 for details.

3 Command line basics

3.1 Files and folders manipulation

To **create a file** (normally this command changes is meant to change the date of a file, but it creates an empty file if it does not exist) : `touch <filename>`

You can send the result of a command into a file by typing `cmd > <filename>`

For example :
```
echo "hello" > thatsme1
whoami > thatsme2
```

The file called "thatsme" now contains the result of the command (`whoami` displays the name of the current user).

To **display the content of a file**, several commands can be used :

`cat` displays the contents of one or more files to screen (concatenate)

`tac` same as cat but reverse

`more` paginates output (stops and wait, next with spacebar)

`less` paginates output (scroll with arrows)

Practice : experiment the commands, display the output of two files at once.
Display the content of a long file (for example `#less /var/log/user.log`)

In addition to the commands `cat less more` there are two useful commands that are useful to **display the beginning or the end** of a file : `head` and `tail`

Example : copy /etc/passwd to the root user home folder and show 10 first lines or 5 last lines

```
#cd ~
#cp /etc/passwd .
#head -10 passwd
#tail -5 passwd
```

The command `tail` is particularly useful to display the most recent entries of log files. These two commands can be combined to display for example the 14th line of the file :

```
#head -14 passwd | tail -1
```

It is also possible to **sort the content of a file** with the command `sort`. In order to test this command,

we need to create a file with several lines including a duplicated line.

```
$echo david > file1
$echo bernard >> file1
$echo arthur >> file1
$echo bernard >> file1
```

Note that >> appends a file while > would overwrite any existing content.

```
$sort file1                compare the output with cat
$sort -u file1             displays unique entries only
```

To **create a folder**, type the command : `mkdir <folder>`
It is possible to create to folders at once :

- at the same level: `mkdir <folder1> <folder2>`
- nested: `mkdir -p folder1/folder2`

The command to **delete a file or a folder** is `rm <file_or_folder>`
Note that a folder must be empty. If not, you have to use the -R option (Recursive) to delete the folder AND its content. When deleting from the command line, files cannot be retrieved. Examples:

```
rm -R <foldername>
rm -R file*
```

(* is the wildcard for any number of characters, while ? matches ONE character only)

The `cp` command lets you **copy a file or a folder** : `copy : cp <source> <destination>`.
Example:

```
cp -R folder1 folder2      (copies folder1 inside folder2)
```

`mv` is the utility to **rename or move files or directories**. If the destination does not exists, it moves and renames with the destination name. Renaming is equivalent to moving to the same location under a new name. Examples:

```
mv oldname newname
mv folder1/folder2 folder2/folder3
mv ../folder1 .            (moves folder1 located one level up to current folder)
```

Exercise :

create a folder 2008 and a folder 2009

in each folder; create subfolders for 3 first months

in each subfolder create a file

rename 2008 as 2009 and 2009 as 2010

delete march 2010

create folder save in /tmp

copy 2009 and 2010 in /tmp/save

3.2 Help and system information

You can **display the system information** (version and name of the Linux distribution) using :

`uname -r` (kernel version : always even numbers. Odd numbers are test versions)

`cat /etc/issue` (system info: Debian GNU/Linux 6.0)

To **get help** ("manual") about a command, type `man <command>`

If you forget a command, you can search for a manual that contains a word using `man -K <word>` or alternatively `-k` to get a list of manual pages with that keyword in their short description only.

Terminology : in a command like `man -K [-M path] keyword`

`man` is the command, `-K` is the option or switch and `keyword` is the argument. Options and arguments are both parameters. Square brackets are used to indicate [optional] parameters. The argument is sometimes provided between < and > signs.

3.3 Text edition : introduction to vim

vim stands for VI iMproved (vi = Visual Interface) : a text editor using commands as opposed to menus like in windows' notepad. Other Linux text editors include *nano* (^ corresponds to ctrl) and *emacs*.

To install vim, type : `#apt-get install vim`

To open a file, type: `vim <filename>`

Vim works with different **modes** which are activated by typing specific keys:

normal	ESC
visual	vV
insertion	iaoOAI
command	:

Commands include :

to simply quit	`:q`
to force quit	`:q!`
to save and quit	`:wq`
to force save-quit	`:wq!`

! can be used in conjuction with w to save a file which is read-only. It is strongly recommended to always cleanly exit vim using `:wq` otherwise there remains a swp temporary file.

Visual mode allows you to highlight and copy some text:

y	copy
wy	copy word (position cursor on 1st character)
yy	copy line

The following are **insertion** commands :

a	Append to the right of the cursor
A	Append at the end of the line
i	Insert text to the left of the cursor
I	Insert text to the left of the first non-blank character on current line
o	Open a new line below the current line and insert text
O	Open a new line above the current line and insert text

ESC takes you to the **normal** mode from which you can for example:

w	Move to the start of the next word
e	Move to the end of the next word
^	Move to the first word of the current line
$	Move to the end of the line
x	Delete the character under the cursor (delete=cut, p=paste)
dw	Delete from the current position to the end of the word
dd	Delete the current line.
u	Undo the last command
U	Undo all change to the current line

Vim includes a **tutorial**, type : `vimtutor` (and read until 4.2 included)

Practice : enable highlighting and line numbering options. Open or create a file, type the commands:

```
:syntax on
:set nu
```

From the normal mode, type i to enter the “insertion” mode in which you can type some text. To save, type ESC to return to normal mode and the command `:wq` (w= write and q=quit).

To make the above options permanent, we need to create a configuration file in user's home folder :

`vim .vimrc` (we create a hidden file by starting its name with a dot)

To enable colour and line numbering option for the root user, copy .virmc to the root user home folder

```
$su root
#cp .virmc ~
```

To **search and replace**: `:%s/<word to search>/<word to replace>/gc`
(% specifies all document, s search, g=all instances, c=confirm)

To **open several documents** in the same console (split viewport vertical or horizontal) type `:sv` or `:sp` The commands can include a filename. To toggle windows, use CTRL-W

To **open several tabs**, type `:tabnew [filename]` and toggle using `:tabp(revious)` and `:tabn(next).` Close the tab using the command `:tabc(lose)`

For **fun**, type `help 42` and `:help!`

4 Managing users and groups

4.1 Listing users and user parameters

The list of users is contained in the file **/etc/passwd** which you can open with vim for viewing but not for editing (best done via specific commands). The format is the following (: is the separator):

root:x:0:0:root:/root:/bin/bash

john:x:1000:1000:john,,,:/home/john:/bin/bash

- x indicates that the password is encrypted
- 1000:1000 is the uid:gid (user ID and primary Group ID) for the first user. Other users would have 1001, 1002 etc while 0:0 is the uid:gid for the superuser.
- This is followed by the description (in this example the same as the user name)
- The last parameter is the command line interpreter (here the Bash Shell). There are several command line interpreters :

bash	Bourne Again shell (written by Stephen Bourne)
sh	Bourne shell
ksh	Korn shell (Bell Labs)
ash	Almquist shell (used in resource-constrained environments)
dash	Debian Almquist shell - modern replacement for ash in Debian.

In the list of users, some accounts are only used for a service (daemon; bin; ...) for security reasons. Encrypted passwords are stored in **/etc/shadow** (view only as root, also best not to modify this file as this done by commands).

daemon:*:15146:0:99999:7:::

User login name

Password (* means no password is set, user cannot open a session, ! means account is locked.)

Days since Epoch (01.01.1970) of last password change

Days until change allowed

Days before change required

Days warning for expiration

Days before account inactive

Days since Epoch (01.01.1970) when account expires

Reserved

4.2 Managing users and groups

The id command **displays information** about a user, including the groups to which it belongs:

```
id <username>
```

The output is as follows (note the default specific groups for audio, CD etc..)

```
uid=1000(user1) gid=1000(user1)
groups=1000(user1),24(cdrom),25(floppy),29(audio),30(dip),44(video),46
(plugdev)
```

Creating users is done thanks to the command:

```
#useradd -m -s <shell> <username>
```
or
```
#useradd -ms /<shell> <username>
```

Notes:

- the switch -s waits for an argument, for example /bin/bash, so you cannot type -sm)
- by default, the system sets /bin/sh as the shell
- the switch -m creates user home now (instead of at logon)

By default, accounts are locked (check /etc/shadows) until you **define a password** :

```
#passwd <username>
```

To **modify a user**, invoke the command (type `man usermod` for the many options)

```
#usermod <username>
```

When a user is created, it has a **default profile** copied from etc/skel. If you want to add a configuration file to all users, add it to this folder. Example for the vim configuration:

```
#cd /etc/skel
#ls -al
#cp ~/.vimrc .
```

An alternative is to edit /etc/vim/.vimrc and either add a line or to uncomment (remove ") the appropriate line(s). This will affect all users.

To **create, delete or edit a group**, the commands are the following, respectively:

```
#groupadd <groupname>
#groupdel <groupname>
```

```
#groupmod <groupname>
```

The file /etc/group **lists all groups** (while /etc/passwd only shows the primary groups). When creating a user, a group that has the same name is automatically created.

To a**dd users to a group**, use the `usermod` command as follows :

```
#usermod -aG group1,group2 username
```

The switch a stands for "add", g indicates primary group and G secondary group

Example (first create a user2 and testgroup, then add user2 to the groups testgroup, audio and cdrom) :

```
#usermod -aG test,audio,cdrom user2
```

4.3 Command input/output applied to user management

The symbol | is called pipe and **redirects** output of the first command into the second (in the example below it sends the output of `cat` into `grep`). The `grep` command returns all line in the file that contains a pattern. For example:

`cat /etc/passwd | grep <user>`	display only the line about one user

It is possible to define if the line should start (^) or finish ($) by the pattern. These symbols are part of Regular Expressions (regex), a complex but very precise, powerful and flexible method for matching strings of text. For example :

`cat /etc/passwd | grep ^us`	outputs all users who's name stars by "us"
`cat /etc/passwd | grep false$`	outputs users with not shell assigned

The following commands are more example of solutions to check system users and groups (note the possible switches for the "count" command `wc` : -l (line) -w (word) -m (character):

`who | grep root`	shows sessions that are open by root
`who | wc -l`	shows number of open sessions (connected users)
`wc -l /etc/group`	shows the number of groups
`cat /etc/group | wc -l`	same as above

The `cut` command is useful to display only a section of a line. The -d switch stands for "delimiter", which is a colon in the passwd field. The -f specifies the field numbers. The following commands display respectively: only the 3rd field (user ID), the 3rd and the 5th, from the 3rd to the 5th :

```
#cat passwd | grep <user> | cut -d ":" -f3
#cat passwd | grep <user> | cut -d ":" -f3,5
#cat passwd | grep <user> | cut -d ":" -f3-5
```

Exercise:

show number of users : `wc -l /etc/passwd`
show connected users : `users | wc -w` OR `who | wc -l`
show main group id of first user created : `cat passwd | grep x:1000 | cut -d ":" -f4`
show groups of current user: `grep `whoami` /etc/group | cut -d":" -f1`

In the corrections, note the use back-quotes (`)

5 Permissions and ownership

5.1 Understanding permissions in Linux

To display the permissions on a file (or folder), type the command `ls -al.`

```
-rwxrw-rw- 1 user1 user1  721 Jun 21 09:15 file1
drwxr-xr-x  2 user1 user1 4096 Jun 21 19:25 Pictures
```

The first block is for the user, the second for the primary group, the third for others. Every file has an owner (here user1). The letters represent the actual permissions, and each have a corresponding value :

r	read	4
w	write	2
x	execute	1
d	directory	(always has x : a folder must be executable)

Folder default permissions are `755 (drwxr-xr-x)`
File default permissions are `644 (-rw-r--r--)`

5.2 Changing permissions

If you create a file that contains some commands that you want to execute, you will need to make the files executable (a normal file is by default not executable). For example, create file in a user's home folder using vim, and type `whoami` in the file.

- Check the permissions : 644
- Try to execute the file (./<filename>): "permission denied"
- Change permissions: `$chmod u+x <filename>`
- Check the permissions : now rwxr--r-- (744)

This is one possibility to change permissions (r, w, x), by adding (+) or removing (-) the right of the user (u) or the group (g) or others (o) or all (a). Instead of u in this example, you could type g, o or a (all). This is method uses the symbolic notation. Alternatively, you could set the permissions using the numeric notation : `chmod 744 <filename>`

In Linux, a file is created with the creator's permissions, it does not inherit the folder's permissions. To apply the same permissions to all sub-folders, use the recursive switch -R :

`chmod -R 744 <foldername>`

5.3 Ownership

In this example, the owner of file1 is user1 who belongs to the group user1 (see 4.2, when a user is created, it is automatically member of a group of the same name).

-rwxrw-rw- 1 user1 user1 721 Jun 21 09:15 file1

To change owner, type the command `#chown <user:group> <file_or_folder>`
Note that if you only specify one parameter instead of user:group, then user is considered.

Exercise: (see folders. Exercise 3_)

note : check group with id command

6 Directory structure

The majority of these directories exist in all UNIX operating systems and are generally used in much the same way.

```
ls -l /
```

/	primary hierarchy, root directory of the entire file system
/bin	Essential command binaries
/boot	boot files
/dev	devices
/etc	Host-specific system-wide configuration files
/home	Users' home directories
/lib	Libraries
/mnt	mount points for file systems
/media	mount points for removable drives (CD; USB key; ext HD...)
/opt	Optional application software packages (empty by default)
/proc	Virtual filesystem documenting kernel and process status
/root	Home directory for the root user.
/sbin	Essential system binaries (admin commands like ifconfig etc)
/sys	System components
/srv	Site-specific data (empty by default)
/tmp :	Temporary files (cleared after a reboot)
/usr	Secondary hierarchy for read-only user data (majority of utilities and applications)
/usr/bin	Non-essential command binaries (used by all users)
/usr/sbin	Non-essential system binaries (e.g daemons for various network services)
/var	variable files : logs, and application files (spool, databases, website files)

7 Disks and partitions

So that we can add hard-drives, we will first stop the virtual machine by typing either of the following:

```
#halt
#init 0 (shutdown)
#init 6 (reboot)
```

7.1 List partitions and disks

Add two hard disks (default options) from the VM settings. Then restart the VM and type the command that will list the available hard drives and partitions.

```
#fdisk -l
```

```
Disk /dev/sda 16.6 GB, 16610492416 bytes
[...]
Device      Boot        Start       End         Blocks      Id    System
/dev/sda1   *           1           43          340992      83    Linux
/dev/sda2               43          1958        15384577    5     Extended
/dev/sda5               43          737         5573632     83    Linux
/dev/sda6               737         1070        2675712     83    Linux
/dev/sda7               1070        1181        886784      82    Linux swap / Solaris
/dev/sda8               1181        1229        389120      83    Linux
/dev/sda9               1230        1958        5855232     83    Linux
Disk /dev/sdb: 1073 MB, 1073741824 bytes
[...]
Disk /dev/sdb doesn't contain a valid partition table
Disk /dev/sdc: 1073 MB, 1073741824 bytes
[...]
Disk /dev/sdc doesn't contain a valid partition table
```

The first disk is called sda, the second disk sdb etc...

The first partition of the first disk is sda1

The new hard disk are not partitioned.

The column “system” tells the file system used when the partition was formatted. Other file system include : NTFS, FAT32 (VFAT is the Linux equivalent), EXT2, EXT3 (default), EXT4, HFS (Mac), ISO9660 (CDs). Type `man fs` for more information.

7.2 Create the partition

Linux provides a graphical utility to partition a disks:

```
#cfdisk <disk_id>
```
(disk_id = /dev/sd?)

For example:

```
#cfdisk /dev/sdb
```

select new > primary > size: default > write > yes (not y) > quit

check with `fdisk -l` : it now shows as partitioned

7.3 Format the new partition

We will format as EXT3, the default Linux file system.

```
#mkfs.ext3 <partition_id>
```
(partition_id = /dev/sd?#)

For example:

```
#mkfs.ext3 /dev/sdb1
```

Note the following lines in the output :

```
13052 blocks (5.00%) reserved for the super user
```

This allocated space is meant to let root connect and work in case a partition is filled. It is useful for system partition, but can recover space in other cases if ncessary.

```
This filesystem will be automatically checked every 30 mounts or 180 days
```

The system will automatically schedule checks (can be disabled)

7.4 Mount the new partition

So far, the partition is not accessible. We need to create a mount point (a folder), assign the correct permissions and mount the partition in this folder. Mount points should be located in the /mnt folder.

`#mkdir /mnt/disk1`	create mount point
`#chmod 777 /mnt/disk1`	assign permissions
`#mount -t ext3 /dev/sdb1 /mnt/disk1`	mount the partition
`#df -h` or `#mount`	check the partitions that are mounted

We can now access the partition on the new hard disk in /mnt/disk1 and create a file.

7.5 Manage partitions

The command to unmount this partition is either `#umount /dev/sdb1` or `umount /mnt/disk1`
If the partition is not mounted, the folder /mnt/disk1 appears empty.

A partition mounted at the command line will not be accessible at the next boot up until we run the same command again. To automatically mount a partition at boot up, we need to edit a file :

```
#vim /etc/fstab
```

The file lists partitions as follows:

```
<file system> <mount point>   <type>  <options>       <dump>  <pass>
```

add a line with tabs as separators, for example :

```
/dev/sdb1       /mnt/disk1      ext3    defaults        0       2
```

Note that :

- for partitions created at installation, /dev/sdb? is represented by UUID
- there are possible options so that a partition could only used by one user
- errors=remount-ro means if error, mount in read only
- <pass> = mount order

To test, either reboot or ask system to mount partitions in fstab that are not already mounted using :

```
#mount -a
```

Exercice : configure /etc/fstab to automatically mount the second disk

8 Archived and compressed files

8.1 Archiving and compressing

An archive is a collection of files within a single file uncompressed (commonly called a tar file or tarball). The command to archive is tar (which stands for tape archive). It is handy to these archive files the .tar extension. For example, the following command would bundle all the .doc files in the current directory into the alldocs.tar file (switches : c= create, v=verbose, f expects a filename and should be placed last).

```
tar -cvf alldocs.tar *.doc
```

If you omit the file(s) or folder to archive, the command returns a message :

`tar -cvf <archive_name>` Cowardly refusing to create an empty archive

The `gzip` program compresses a single file. One important thing to remember about `gzip` is that, unlike tar, it replaces your original file with a compressed version. The extension .gz is automatically given to the compressed file.

`gzip report.doc`	(compresses report.doc into report.doc.gz)
`gzip -r <directory>`	(compresses every file in the the directory)
`gzip alldocs.tar`	(compresses an archive, creates a tar.gz file)

It is possible to combine archiving and compressing thanks to the `-z` switch of the `tar` command. The switch `P` is optional and indicates that absolute paths should be used. It is recommended to provide a meaningful extension to compressed archives, for example tgz or tar.gz.

```
tar -cPzf <archive_name> <files_or_folders_to_archive>
```

If you provide the files or folder name using the absolute path, and you omit the -P, the systems gives a warning :`Removing leading `/' from member names`

8.2 Checking the archive content

The command tar with the `t` (list) switch allows you to check content of archive. Note that the `tar` command accepts that you omit the – before the switches.

```
tar tvf <archive_name>
```


8.3 Decompressing / restoring

You can decompress the a file that was compressed with the `gzip` command. For example, the command below will return the file report.doc.gz to its original state report.doc.

```
gunzip report.doc.gz
```

The command `tar` with the `-x` switch will restore an archive. Without the P switch, restore will use the relative path, meaning you have to be in the right folder.

```
tar -xPzf <archivename>
```

It is possible to partially restore archive by providing the path.

```
tar -xPzf <archive> </path/of/data/to/restore>
```

Exercises :

1/ Create a user with the correct option to create the home directory : archive, delete and restore it.

2/ create user Bob, and in his home folder create folder1 and add a file in this folder
backup Bob's home folder using tar to /mnt/disk1/backup
delete folder1
restore folder1

Notes about the correction:

`;`	runs two commands in one line, no matter what is the output of the first command
`&&`	runs the second command only if there is no error in first one, based on the return code
`||`	runs second command only if error in first command
`echo $?`	gives return value of last command (0 no error)
`1> /dev/null`	sends the standard output to the null device (“black hole”), i.e. no output displayed
`2> /dev/null`	sends any error to the null device (“black hole”), i.e. no error displayed

see : http://en.wikipedia.org/wiki/Redirection_%28computing%29

9 Managing processes

9.1 Listing running processes

There are various ways to display the list of all running processes :

`top`	lists all processes in realtime as well as system resources
`pstree`	displays process tree (note that init is the first process)
`ps -ef`	displays processes with PPID
`ps -aux`	displays processes with CPU and memory usage

PID = Process ID (numbered by order of creation)
PPID = Parent Process ID

If you want to monitor system resources real time, you can use an application called saidar :

```
#apt-get install saidar
#saidar -c
```

`uptime`	displays how long the system has been running

Exercise : launch vim and find its process ID using the command grep

9.2 Stopping a process

In case an application hangs, you can stop the related process with the command `kill` followed by the process ID :

`kill <PID>`	unless logged as root, you can only kill process that you started

This command sends a signal to a process, by default 15 (TERM) which asks the application to terminate. Should this signal fail, you can use 9 (KILL) which cannot be blocked. To stop all processes with the same name as oppposed to stopping according to user or pid, use the following commands :

```
#killall <process_name>
#pkill <process_name>
```

Exercise : install Lynx (text mode web browser), launch it and kill it

```
#apt-get install lynx
#lynx <ulr>
```

10 Symbolic links

A symbolic link (also called symlink or soft link) is a special type of file that contains a reference to another file or directory. The command to create a symbolic links is the following (note that <link_name> must not be an existing file or folder). It is recommended to use absolute paths. The command `ls -l` shows symbolic link with an arrow pointing to the actual folder or file.

```
ln -s <source_file_or_folder> <link_name>
```

Symbolic links are particularly useful if a partition is running out of disk space and it is fully transparent for the user and for the system alike. For example, suppose a server has a partition for /etc, /lib, /root and /var that has become too small, and another with /home that has plenty of space. It is possible to create a symbolic link so that /var points to /home/var :

stop processes of applications creating logs
```
#cp -pR /var /home/var          ( -p switch to copy keeping ownership)
#rm -R /var
#ln -s /home/var /var
```
restart processes

To remove symbolic, type :

```
rm <link_name>
```

Note that there is no switch -R as it only deletes the links, not the content of actual folder. If you remove or rename the actual folder (source), the symbolic link remains but points to nothing.

11 Schedules jobs

11.1 Examples of configuration

Cron is a time-based job scheduler common to all Unix-based systems. Cron enables users and administrators to schedule jobs (commands or shell scripts) to run periodically at set times or dates. This utility is driven by a configuration file called crontab located in /etc.

In the example below, the script would run on a weekly basis at 6:47am every Sunday.

```
#vim /etc/crontab
 m          h     dom        mon        dow        user       command
47          6     *          *          7          root       /path/script
```

m : minute
h: hour
dom : day of month
mon : month
dow : day of week
* = any

Examples :

5th of June at 13:26

```
m     h     dom   mon   dow   user   command
26    13    5     6     *
```

any Friday 13th, at 13:13

```
m     h     dom   mon   dow   user   command
13    13    13    *     5
```

any working day of the week in July and May

```
m     h     dom   mon   dow   user   command
*     *     *     5,7   1-5
```

Note : it is recommended to point the entry in the crontab file to a script rather than typing the actual command. This also allows for testing the script prior to creating the scheduled job.

11.2 Application to automated backup

Below is how you would plan a backup of the /home folder to /mnt/disk1 using the command tar:

```
#cd /root
#vim home_backup
```

type the command : `tar cPzf /mnt/disk1/backup/home.tgz /home`

make this file executable (chmod u+x)
test script before configuring crontab
add script to crontab and configure it 5 or 10 minutes ahead of the current time

45 10 * * * root /root/home_backup

Doing so however means that each daily backup will overwrite the previous one. This can be avoided by including the date in the file name using the command `date` between back-quotes :

```
/mnt/disk1/backup/`date +%Y_%m_%d`_home.tgz
```

Swapping `%Y_%m_%d` is possible but in general, providing the year first makes it easier to sort as files will appear by default in the actual order when listed with the command `ls`.

For a more elaborate script, see the file cron_ex

12 System event logs (syslog)

Syslog is a standard for logging program messages. The program rsyslog is available for a number of Unix systems and Linux distributions, among others. It uses the configuration file rsyslog.conf. Logs are store in /var/log

```
ls -l /var/log
-rw-r----- 1 root        adm             0 Jun 21 18:36 mail.err
-rw-r----- 1 root        adm             0 Jun 21 18:36 mail.info
-rw-r----- 1 root        adm             0 Jun 21 18:36 mail.log
-rw-r----- 1 root        adm             0 Jun 21 18:36 mail.warn
```

The directory listing for the `mail` service show that its logs are set with different security levels.

0 - Emergency (emerg)
1 - Alerts (alert)
2 - Critical (crit)
3 - Errors (err)
4 - Warnings (warn)
5 - Notification (notice)
6 - Information (info)
7 - Debug (debug)
* - any

These security levels are set in the configuration file along with the "facilities" which loosely relate to system processes. Some facilities log all types of events into one log file.

```
#vim /etc/rsyslog.conf          (go to line 54)
kern.*       -/var/log/kern.log          (any kernel related event is logged in kern.log)
mail.err      /var/log/mail.err          (mail errors logged in a specific file)
```

auth - authentication (login) messages
cron - messages from the memory-resident scheduler
daemon - messages from resident daemons
kern - kernel messages
lpr - printer messages
mail - messages from Sendmail
user - messages from user-initiated processes/apps
syslog - messages from the syslog process itself

You can display the last 20 lines of some of the files, for example the authentication log :

```
#tail -20 /var/log/auth.log
```


13 Installing applications

13.1 Using apt-get and the repositories

Every Linux distribution has its own way of packaging software. For example, Red Hat uses .rpm packages to install software and Debian uses .deb packages. These packages are made available in a public repository where they are organized in a special directory tree. Depending on the location / country provided at installation, the system will set certain repositories as preferred. In a typical installation, you would first if necessary add a repository, then update the packet list, update the packets and only then install:

`#vim /etc/apt/sources.list`	list repository in use
`#apt-get update`	update the packet list (resynchronize)
`#apt-get upgrade`	update packets
`#apt-get install <packetname>`	install a packet
`#apt-cache search <name>`	search packets
`#apt-get remove <packetname>`	remove a packet (uinstall)
`#apt-get remove --purge <packetname>`	remove a packet including config files

The graphical interface (synaptic) also uses `apt-get`. Note that there an alternative to `apt-get` called `aptitude`. It is recommended to always use either `apt-get` or `aptitude`.

I you install, then uninstall, and later reinstall, the system does not need to download again, it will the packets stored locally. It is also possible to manually download .deb files but this is not recommended as you will then have to manually update the applications.

`wget` download item from webpage
(a gpg key a identification key system people use to "sign" files so you can check their authenticity)

Example 1: installation of wine (allows running windows applications on linux)

#apt-get install wine

Information about Wine :	http://www.winehq.org
Help installing Windows application via Wine :	http://www.playonlinux.com

Launch Play-On-Linux from the graphic interface (in the folder “games”). It will update itself and present

a list of applications. It depends on Wine (either install first; or it will install it).

Example 1: installation of VMware tools (see script "tools")

```
rm -R /tmp/VM*
rm -R /tmp/vm*

mkdir /home/temp
cd /home/temp
scp <user>@<srv_IP>:VMwareTools-8.4.2-261024.tar.gz .   (copy from network)
tar xzf VmwareTools-8.4.2-261024.tar.gz                 (extract)
cd vmware-tools-distrib

apt-get update
apt-get install -y gcc linux-headers-`uname -r`         (install dependencies)

./vmware-install.pl –default                            (install)
```

Note : `--default` means it will automatically take all default options

13.2 Installing using source files

Generally you would get the source files for a Linux software in the tarball format (.tgz) This file has to be uncompressed into any directory using tar command. The software installation is then done in 3 steps : configure, make, make install.

The **configure** script basically consists of many lines which are used to check some details about the machine on which the software is going to be installed. The main job of the configure script is to create a file called ' Makefile '

The **make** utility compiles all your program code. It creates the executables using the directions present in the Makefile and proceed with the installation.

But when type **make install** (you run make with install as the parameter), the make utility searches for a label named install within the Makefile, and executes only that section of the Makefile. The install section happens to be only a part where the executables and other required files created during the last step (i.e. make) are copied into the required final directories on your machine.

Note that this procedure is more complex than using `apt-get` or `aptitude`, and only applies to software not available from a repository (e.g. recent updates, test versions) and for other specific needs as well as Slackware-type distributions.

Example : aMSN

From the aMSN website, download the tarball source files or use the `wget` command after positioning yourself in the appropriate directory. Then extract the files (no need for -P in the `tar` command for web downloads).

```
#wget http://mesh.dl.sourceforge.net/project/amsn/amsn/0.98.4/amsn-0.98.4-src.tar.gz
#tar -xzf amsn-0.98.4-src.tar.gz
```

Before installing aMSN, you need to install required packages (dependencies):

```
#apt-get install -y linux-`uname -r` make gcc g++ tcl8.5-dev tk8.5-dev libpng-dev libjpeg-dev
```

Note that linux-headers, `make`, `gcc` and `g++` are basic requirements. Linux headers are the development files that are nescessary to build / compile kernel modules. `gcc` and `g++` are compilers. The other elements are specific to aMSN.

Once these dependencies are installed, you can compile and install the software :

```
#cd amsn-0.98.4
#./configure && make
#make install
```

Note : if you get an error, uninstall previous dependencies : `#apt-get remove tcl8.4 tk8.4`

14 Appendix : additional commands

Display disk usage :

`du-sh` disk used

`df-h` disk free

(s = sum otherwise would list all files)

Finding an item :

`find <path> -iname <item_to_find>`

(iname = not case sensitive)

can use wildcards * and ?

