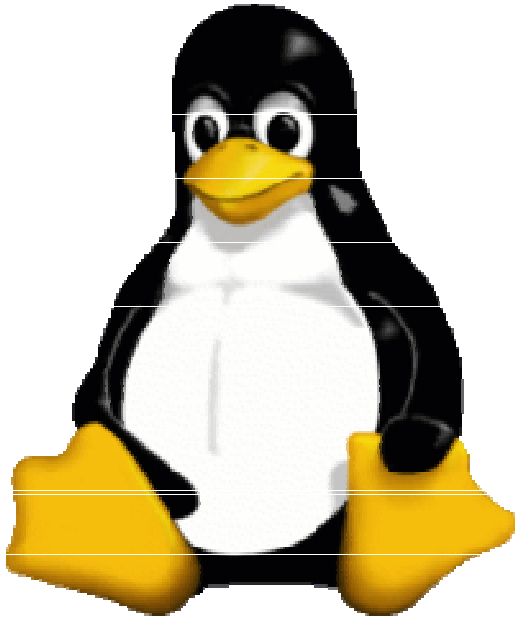
Introduction to Linux

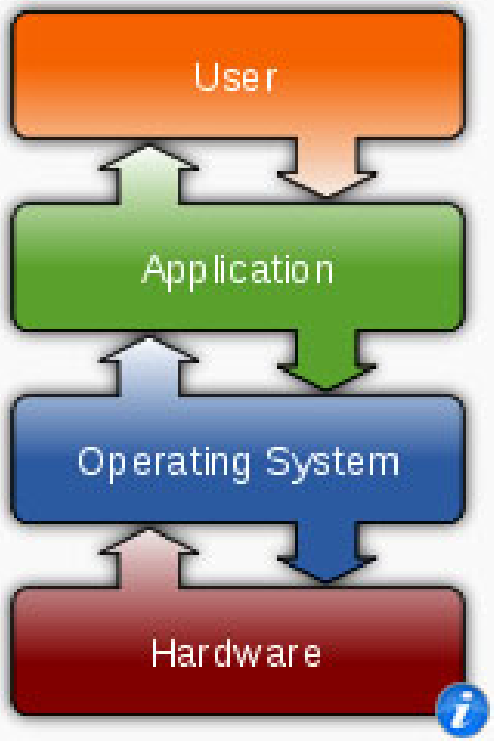
Credit to:



- Mag Selwa
- ICTS Leuven
- https://hpcleuven.github.io/Linux-intro/
- Don Johnson @BU

What is Linux?

It's an Operating System

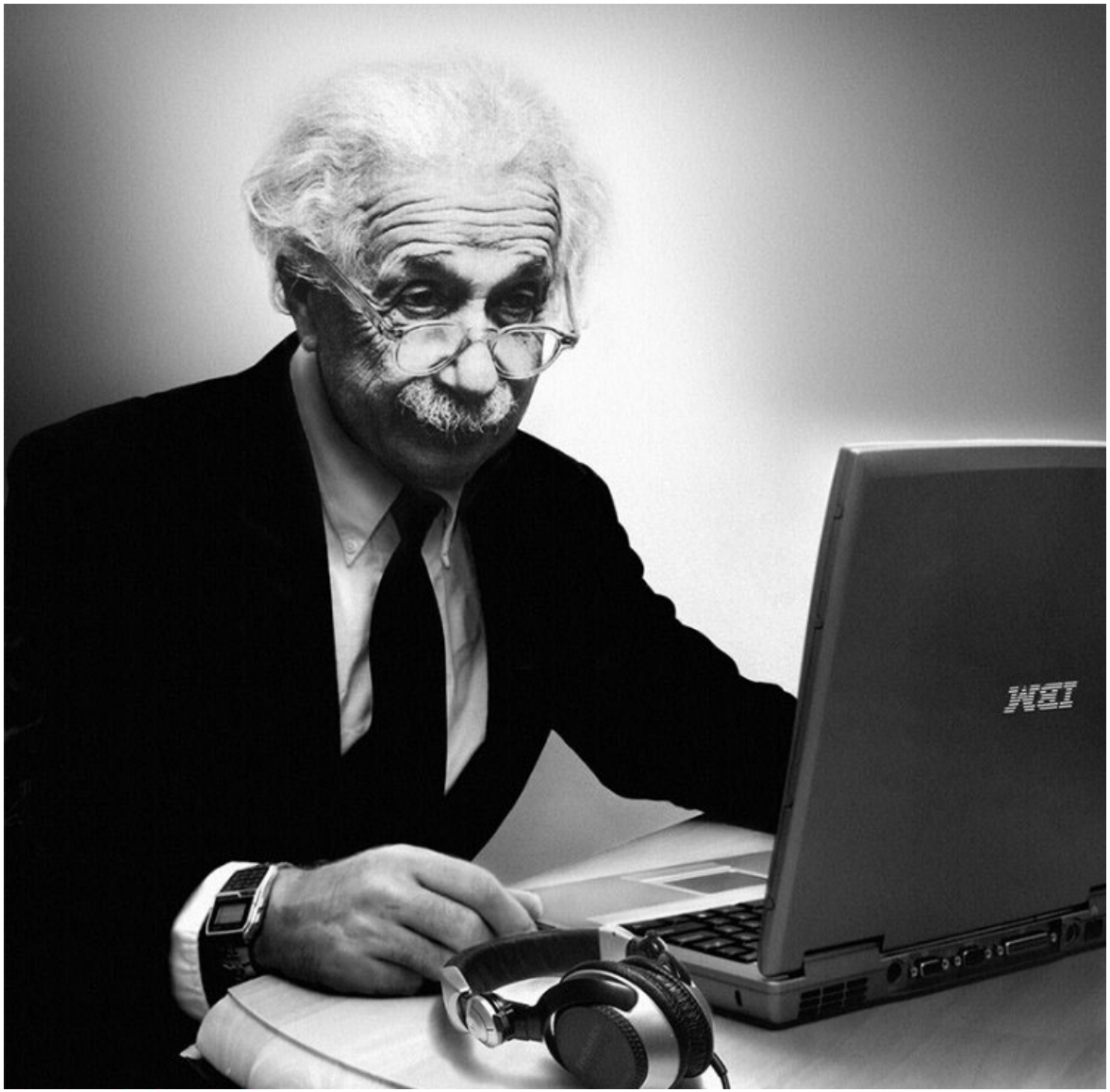


Common features

- Process management
- Interrupts
- Memory management
- File system
- Device drivers
- Networking (TCP/IP, UDP)
- Security (Process/Memory protection)
- I/O

What is Linux?

- The Most Common O/S Used By BU Researchers When Working on a Server or Computer Cluster



What is Linux?

- Linux is a Unix clone written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net.
- Unix is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs.
- Linux and Unix strive to be POSIX compliant.
- 64% of the world's servers run some variant of Unix or Linux. The Android phone and the Kindle run Linux.

The Linux Philosophy

The *Nix Philosophy of Doug McIlroy

(i) Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new features.

(ii) Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.

(iii) Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

Websites?

- **http://www.howtogeek.com/tag/linux/**
  - Includes help, tutorials, tips and how-to guides for Linux.
- **http://www.tldp.org**
  - The Linux Documentation Project
- **http://www.lwn.net**
  - Linux Weekly News:Covering the Linux and free software communities since 1998.
- **http://www.linuxjournal.com**
  - The monthly magazine of the Linux community, promoting the use of Linux worldwide.

Try it?

- Install virtual machine software
  - Vmware player
  - http://www.vmware.com/products/player/
  - Virtualbox
  - https://www.virtualbox.org/
- Install a linux distribution
  - Ubuntu http://www.ubuntu.com/
  - CentOS https://www.centos.org/
  - Opensuse http://www.opensuse.org/
  - …

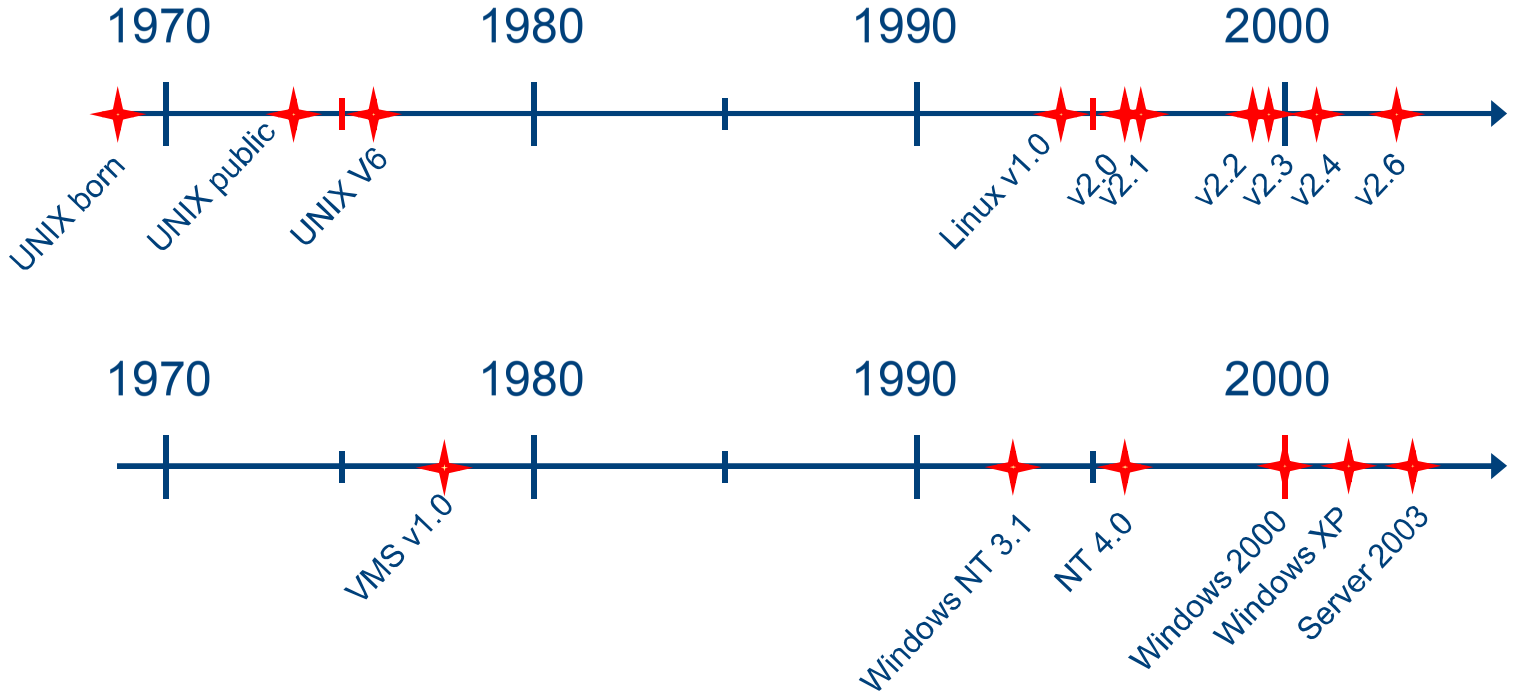
Knoppix can be a good solution

You **do not need to install** Knoppix on harddisk

- So it can be used in Demo of linux or software on Linux,
- So you need extra Linux machine lab in a few minutes,
- No extra space on harddisk on old PC's, just use Knoppix,
- Got a new laptop, just boot Linux on it.

Windows and Linux

- Both Linux and Windows are based on foundations developed in the mid-1970s
- Both DOS, MAC and UNIX are proprietary, i.e., the source code of their kernel is protected
- No modification is possible without paying high license fees



What is Linux?

Linux + GNU Utilities = Free Unix

- Linux is an O/S core written by Linus Torvalds and others AND

- a set of small programs written by Richard Stallman and others. They are the GNU utilities.

http://www.gnu.org/

Some history

GNU project:

- Established in 1984 by Richard Stallman (goal: software should be free from restrictions against copying or modification in order to make better and efficient computer programs),
- GNU is a recursive acronym for “**G**NU's **N**ot **U**nix”,

- Aim at developing a complete Unix-like operating system which is free for copying and modification,
- Companies make their money by maintaining and distributing the software, e.g. optimally packaging the software with different tools,
- Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed

Linux: some history

- UNIX: roots in Bell Labs (AT&T)
- 1985 Free Software Foundation (FSF) founded by Richard Stallman. Along with other programmers creates the tools needed to make a UNIX compatible OS
- 1985 Professor Andy Tannenbaum creates a UNIX like operating system based on System V Unix for the IBM PC & PC/AT computers. It is called Minix.
- 1989 Richard Stallman releases GPL and GNU software but lacks a free kernel.
- 1991 Building on the concepts in Minix, Linus Torvalds (Finnish college student) develops Linux along with help from other users on the web.

Linux Has Many Distributions

Linux Has Many Distributions

- BU uses CentOS in its Linux cluster which is a free version of RedHat Enterprise Linux with the trademarks removed

Operating system?

- Strictly speaking Linux refers to the kernel
- GNU/Linux more accurately describes the Operating System. Linux Kernel combined with GNU utilities and libraries
- Distribution – GNU/Linux bundled with other applications. Examples Red Hat Linux, Debian, Ubuntu, Suse, Knoppix, etc.
- Distributions can be compiled and maintained by an individual or corporation. Can be small (single floppy disk) or span several CD/DVDs.

www.distrowatch.com for more information

Linux Kernel

- File Management
  - Controls the creation, removal of files and provide directory maintenance
  - For a multiuser system, every user should have its own right to access files and directories
- Process Management
  - For a multitask system, multiple programs can be executed simultaneously in the system
  - When a program starts to execute, it becomes a process
  - The same program executing at two different times will become two different processes
  - Kernel manages processes in terms of creating, suspending, and terminating them
  - A process is protected from other processes and can communicate with the others

Linux Kernel

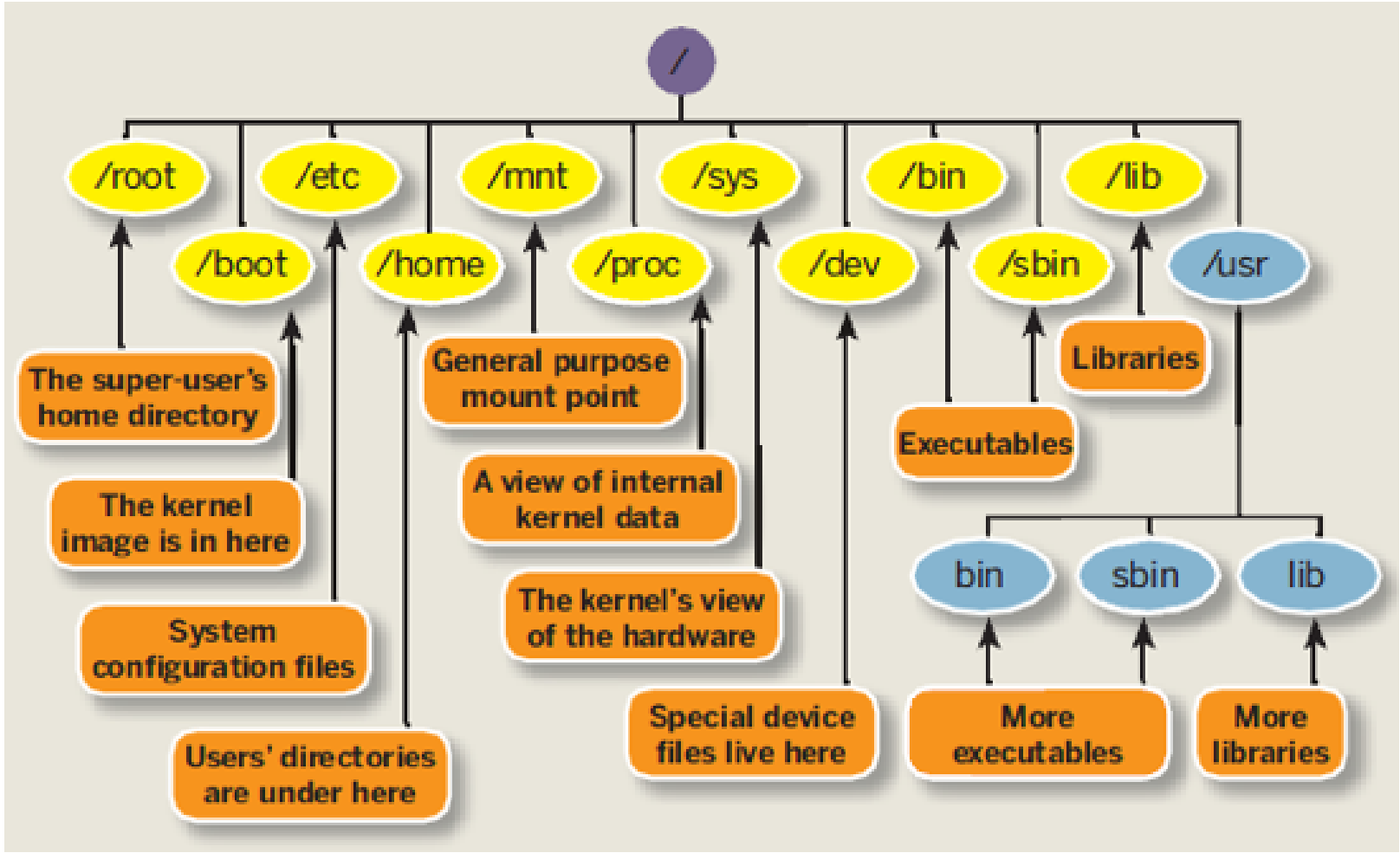
- Memory management
  - Memory in a computer is divided into main memory (RAM) and secondary storage (usually refer to hard disk)
  - Memory is small in capacity but fast in speed, and hard disk is vice versa
  - Data that are not currently used should be saved to hard disk first, while data that are urgently needed should be retrieved and stored in RAM
- Device drivers
  - Interfaces between the kernel and the BIOS
  - Different device has different driver

User login

- Linux is a multiuser OS ,
- Allows multiple users to use the resource of a computer at the same time
- Every user needs to login the system with the password provided to identify their right in using the resource
- Requires for both client-server based system or desktop

Linux
Server

Linux File System



Source: http://linuxsuperuser07.blogspot.be/2011/09/rhel-6-file-system.html

Linux File System

Not imposed by the system. Can vary from one system to the other, even between two GNU/Linux installations!

`/`	Root directory
`/bin/`	Basic, essential system commands
`/boot/`	Kernel images, initrd, configuration files
`/dev/`	Files representing devices
`/etc/`	System configuration files
`/home/`	User directories
`/lib/`	Basic system shared libraries
`/media/`	Mount points for removable media

Linux File System

`/lost+found/`	Corrupt files the system tried to recover
`/mnt/`	Mount points for temporarily mounted filesystems
`/opt/`	Specific tools installed by the sysadmin `/usr/local/` often used instead
`/proc/`	Access to system information
`/sbin/`	Administrator-only commands
`/sys/`	System and device controls
`/tmp/`	Temporary files

The Unix filesystem structure is defined by the Filesystem Hierarchy Standard (FHS):
http://www.pathname.com/fhs/pub/fhs-2.3.html

http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

Linux User Interface

Traditional Linux (Unix also) uses command-driven interface (or text-based interface)

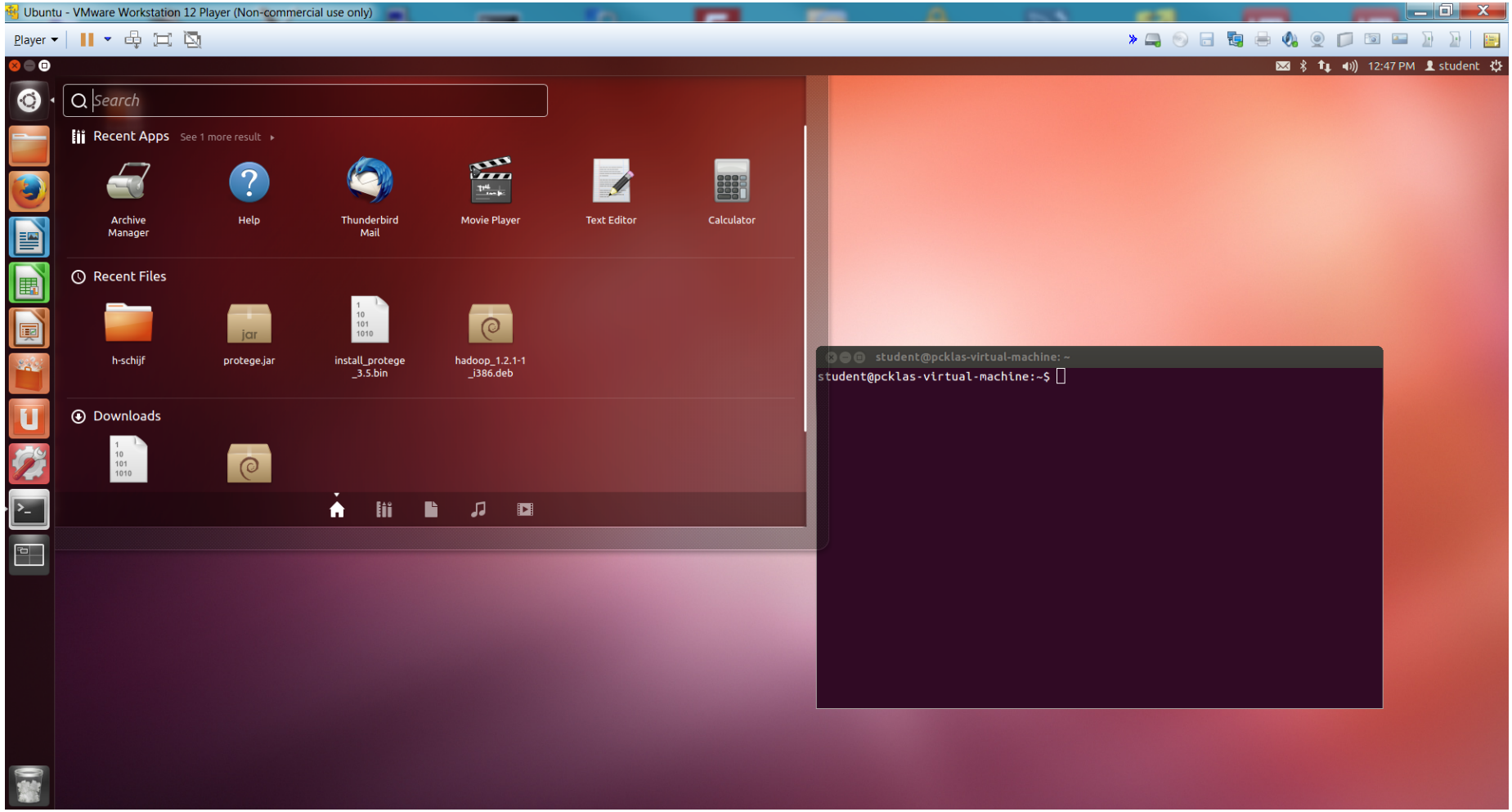
- User needs to type lines of command to instruct the computer to work, similar to DOS
- Advantage: fast in speed. Very few resource is required for its implementation
- Disadvantages: user needs to type, hence can easily make error. Besides, user needs to memorize all commands
- Suitable for expert users and for the systems that interaction with user is not frequent, such as servers

Linux User Interface

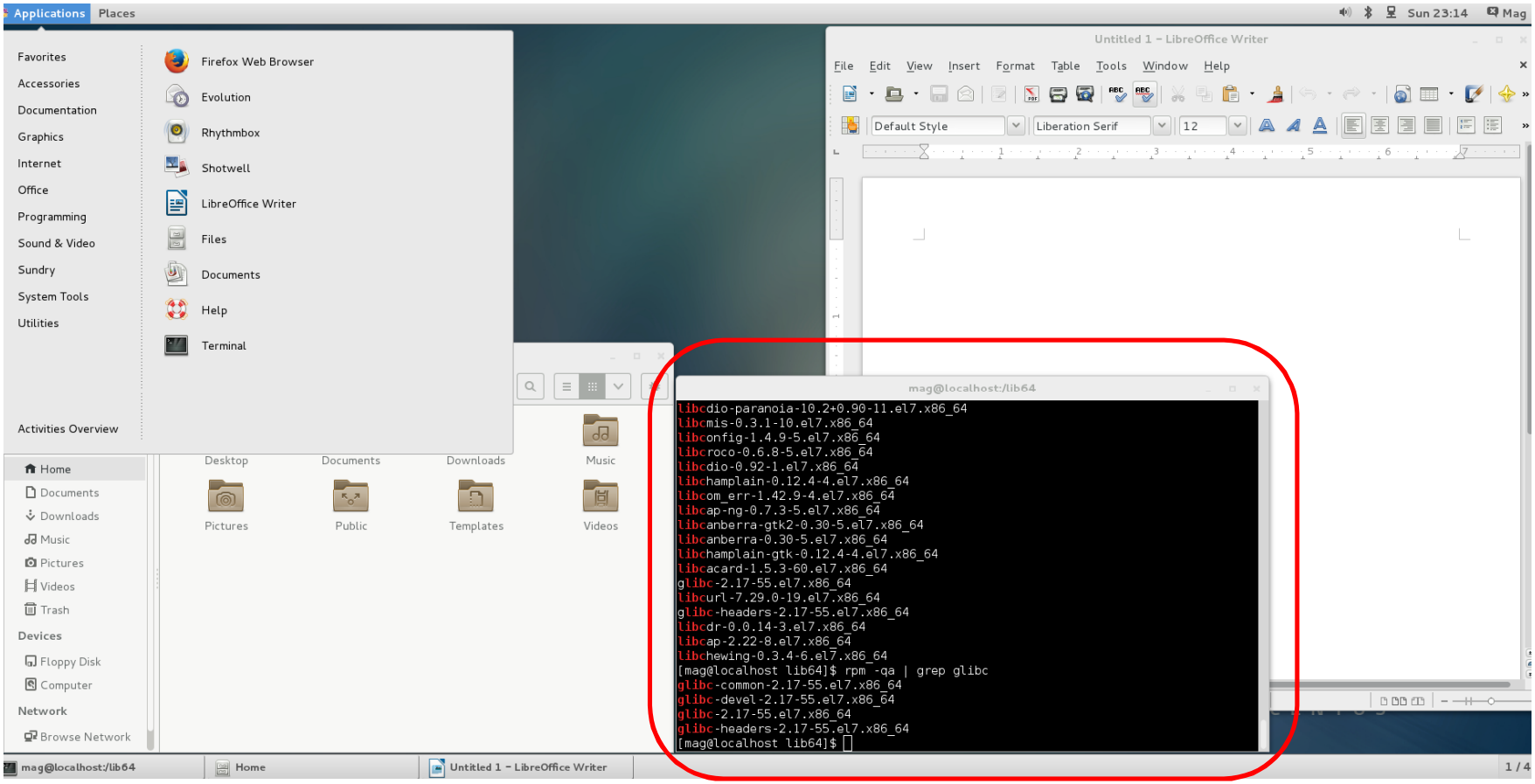
By adopting the X-Window technology, graphical user interface (GUI) is available for Linux:

- Uses pointing devices (e.g. mouse) to control the system, similar to Microsoft's Windows
- Provide menu-driven and/or icon-driven interfaces
  - menu-driven: user is provided with a menu of choices. Each choice refers to a particular task
  - icon-driven: tasks are represented by pictures (icon) and shown to user. Click on an icon invokes one task
- Advantages: No need to memorize commands. Always select task from menus or icons
- Disadvantages: Slow and require certain resource for its implementation
- Suitable for general users and systems, such as PC

Linux User Interface



Linux User Interface



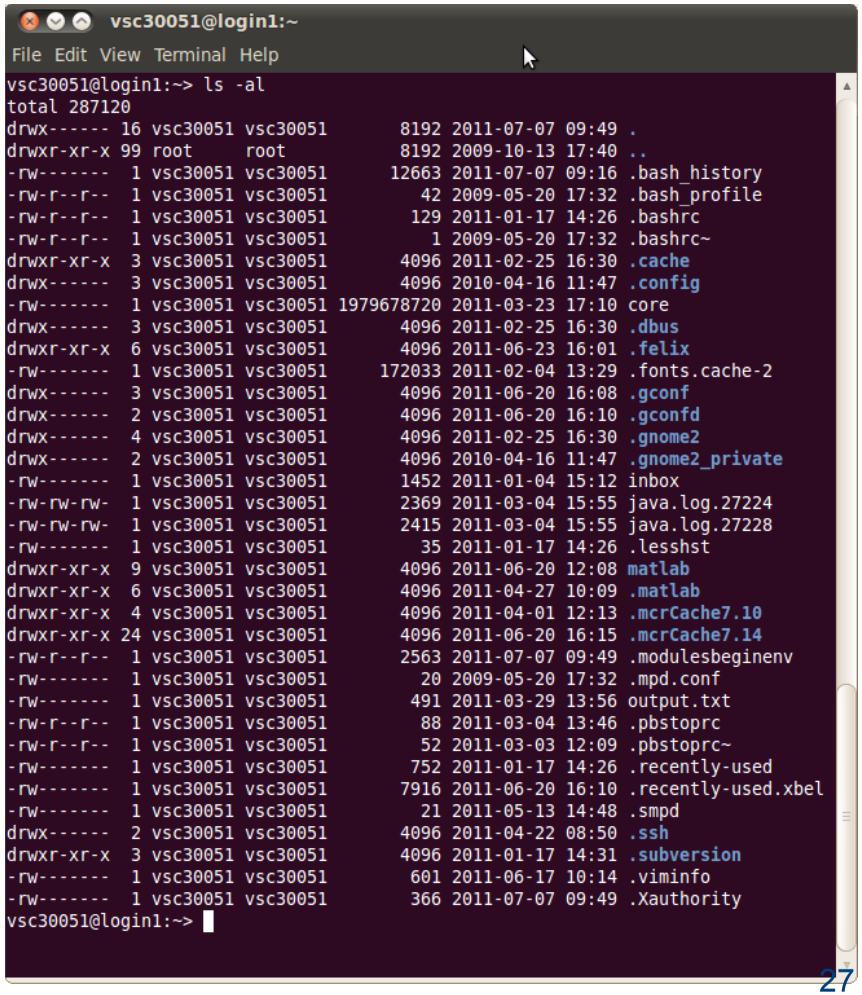
Linux text based interface: The shell

- Is the command line interpreter: a program that accepts input from a user (e.g. a command) and performs the requested task.
- The shell's prompt identifies the type of shell being used. There are two basic types of shell prompts:
  - **$** Normal user shell (may also be **%** or **>** on some systems)
  - **#** Root user shell (login as root or `$ su`)

    !!! Warning: The root user can do anything on Linux systems. It is important to know when you are working as root to prevent accidental damage to your system.
  - The way to identify the user: `$ whoami`

Command Line

- Text mode
- Provides a way to control the computer via the keyboard.
- Is common use on HPC
- Is somewhat archaic, but very powerful.
- Can be much quicker than a GUI.
- Implicitly assumes that you know what your are doing
- Don't be scared!
- If anything goes wrong, you can stop the command with **`Ctrl+C`**



Important rules on the command line

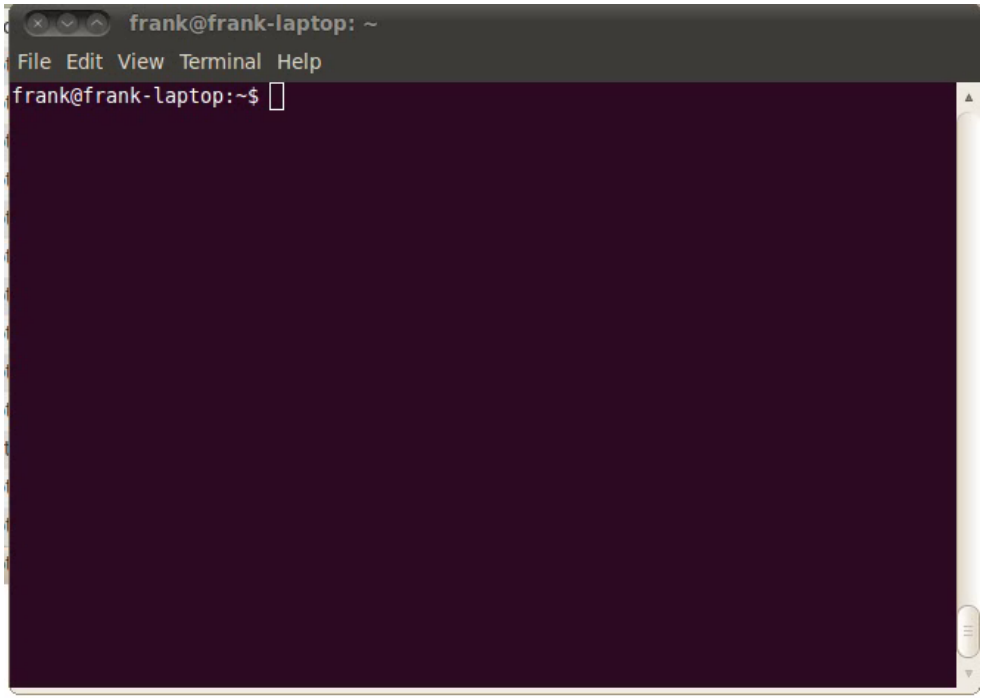
1. Linux systems are case (and space) sensitive.
   - `MyFile` is not same as `myfile`
2. There is no "recycle bin" or "trash can" when working in the command line environment. There might be one for GUI.
   When files are deleted on the command line, they instantly disappear forever.
3. You should always practice new commands on a test system that is not used in a production environment. This minimizes the chances of an accident that can take down an important system

Important rules on the command line

- “\” vs. “/”:
  - In Linux, the “/” is the directory separator, and the “\” is an escape character.
  - In Windows, the forward-slash “/” is the command argument delimiter, while the backslash “\” is a directory separator
- Filenames:
  - In Linux, there is no such thing as a file extension. Periods can be placed at any part of the filename, and “extensions” may be interpreted differently by all programs, or not at all.
  - Windows uses the “.extension” filename convention, (e.g. FILENAME.TXT).

Getting help: command built-in

- Help on most Linux commands is typically built into the command themself
- These flags usually look like "`-h`" or "`--help`".
- `$ ls --help`



Getting help: man pages

- Best source of information can be found in the online manual pages, "**man pages**" for short.
  type "`man command`".

  ```
  $ man grep
  ```

- Tips:
  - To search for a particular word (e.g. file) within a man page, type "`/word`".
  - To quit from a man page, type the "`q`" key.
  - If you do not remember the name of Linux command and you know a keyword relating to the command, search the man pages with the `-k`

    ```
    $ man -k control
    ```

Getting help: info pages

- Info pages are similar to man page, but instead of being displayed on one long scrolling screen, they are presented in shorter segments with links to other pieces of information.
- Access with the "`info`" command

  ```
  $ info ls
  ```

- Tips:
  - To quit from a info page, type the "`q`" key.
  - Type "h" to get more help on the info

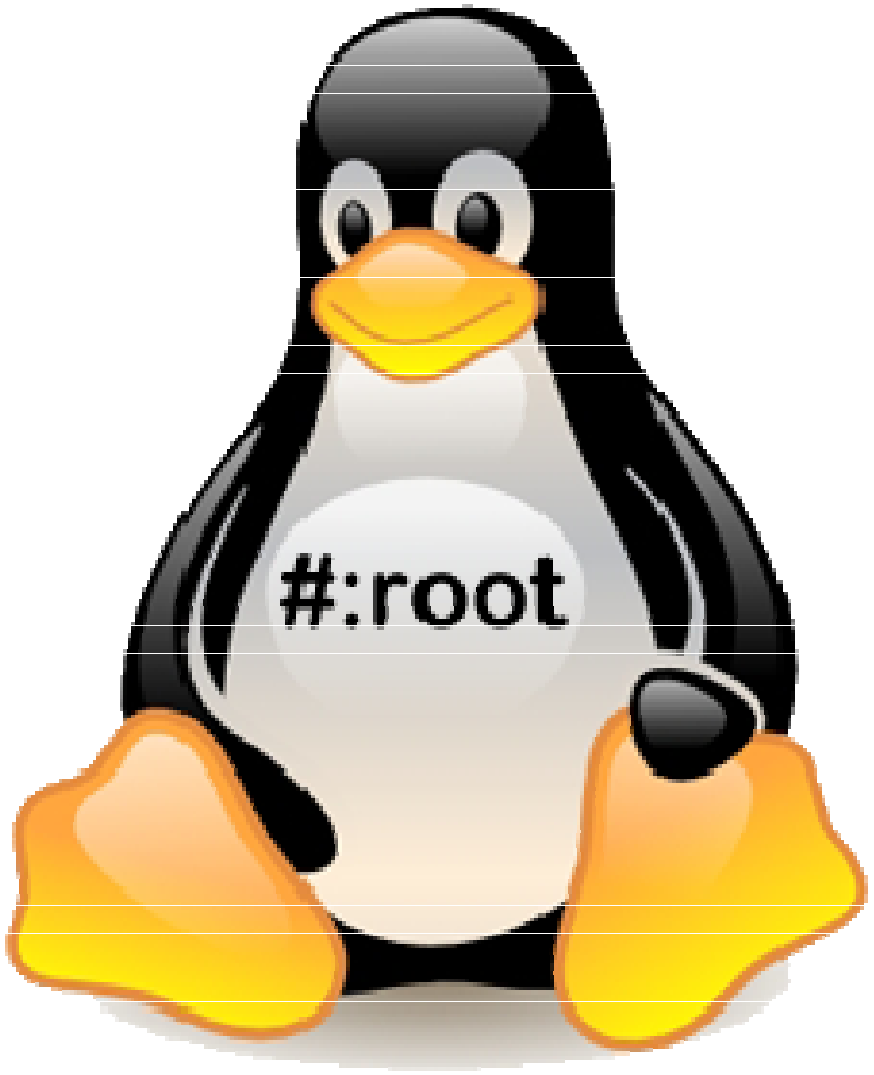
Getting help

- bash has a built-in help facility available for each of the shell *builtins*.

```
$ help cd
$ help pwd
```

- `whatis` displays a very brief description of a command

```
$ whatis pwd
```



The root user is the master

Using the root account

- If you have the root password:
  `$ su` - (switch user)
- In modern distributions, the sudo command gives you access to some root privileges with your own user password.
  Example: `$ sudo mount /dev/hda4 /home`
- root user privileges are only needed for very specific tasks with security risks: mounting, creating device files, loading drivers, starting networking,
  changing file ownership, package upgrades...
- Check the prompt **#**

Brief list of commands

- **Network:** ssh, scp, ping, telnet, nslookup, wget
- **Shells:** BASH, TCSH, alias, watch, clear, history, chsh, echo, set, setenv, xargs
- **System Information:** w, whoami, man, info, which, free, echo, date, cal, df, free, man, info
- **Command Information:** man, info
- **Symbols:** |, >, >>, <, &, >&, 2>&1, ;, ~, ., .., $!, !:<n>, !<n>
- **Filters:** grep, egrep, more, less, head, tail
- **Hotkeys:** <ctrl><c>, <ctrl><d>
- **File System:** ls, mkdir, cd, pwd, mv, ln, touch, cat, file, find, diff, cmp, /net/<hostname>/<path>, mount, du, df, chmod, find
- **Line Editors:** awk, sed
- **File Editors:** vim, gvim, emacs –nw, emacs

System Information

- After you connect, type
  - **shazam**
  - **whoami**
  - **hostname**
  - **date**
  - **cal**
  - **free**
- Commands have three parts; *command*, *options* and *parameters*. Example: **cal –j 3 1999**. “cal” is the command, “-j” is an option (or switch), “3” and “1999” are parameters.
- Options have long and short forms. Example:
  - **date –u**
  - **data --universal**

What is the nature of the prompt?
What was the system’s response to the command?

Command History and Simple Command Line Editing

- Try the **history** command
- Try **<Ctrl><r>** (only works in BASH shell)
- Choose from the command history by using the up ↑ and down ↓ arrows
- What do the left ← and right → arrow do on the command line?
- Try the <**Del**> and <**Backspace**> keys

Help with Commands

- Type
  - **hostname –-help**
  - **man hostname**
  - **info hostname** (gives the same or most information, but must be paged)
- And “Yes,” you can always Google it

Connect Commands Together with the Pipe Symbol “|” and Using Filters

- The pipe “|” feeds the OUTPUT of one command into the INPUT of another command. Our first example will use the pipe symbol to filter the output of a command. Try:
  - **w**
  - **w | grep ‘root’**
  - **ps -e -o ruser,comm | grep 'tut‘**
- The **ps** command is using both “options (dash)” and parameters
- Try both “**man grep**” and “**info grep**”. See the difference?

Editing the Command Line with Emacs Keys (see emacs-editing-mode.pdf)

- **<Ctrl-a>** go to beginning
- **<Ctrl-e>** go to end
- **<Alt-f>** forward one word
- **<Alt-b>** back one word
- **<Ctrl-f>** forward one character
- **<Ctrl-b>** back one character
- **<Ctrl-d>** delete character
- **<Alt-d>** delete word
- **<Ctrl-u>** delete from cursor to beginning of line
- **<Ctrl-k>** delete from cursor to end of line

See emacs-editing-mode.pdf and emacs-editing-mode-short.pdf

Go to through command history in shell and practice editing.

Modifying the Linux File System

- More useful commands
  - **cd** (also takes you to your home directory like cd ~)
  - **mkdir test**
  - **echo ‘Hello everyone’ > test/myfile.txt**
  - **echo ‘Goodbye all’ >> test/myfile.txt**
  - **less test/myfile.txt**
  - **mkdir test/subdir1/subdir2** (FAILS)
  - **mkdir -p test/subdir1/subdir2** (Succeeds)
  - **mv test/myfile.txt test/subdir1/subdir2**
  - **rmdir test** (FAILS)
  - **rm –Rv test** (Succeeds)

The List Command

- Useful options for the “**ls**” command:
  - **ls -a** List all file including hidden file beginning with a period “**.**”
  - **ls -ld *** List details about a directory and not its contents
  - **ls -F** Put an indicator character at the end of each name
  - **ls –l** Simple long listing
  - **ls –lh** Give human readable file sizes
  - **ls –lS** Sort files by file size
  - **ls –lt** Sort files by modification time

File System Ownership and Permissions

- All files and directories have a individual and a group *ownership.*
- All files and directories have read (r), write (w), and execute (x) *permissions* assigned as octets to the individual owner (u), the group (g) owner and all others (o) that are logged into the system.
- You can change permissions if you are the individual owner or a member of the group.
- Only root can change ownership.

Editing Output Lines With *sed*

- **sed** replaces one substring with another
- **sed** operates on every line in a file or processes every line piped into it
- **sed** matches patterns using *regular expressions* (See regular-expressions.pdf cheat sheet)
- Common regular expression metacharacters:
  - . – any character
  - ? – quantified zero or one
  - * - quantifier none or more
  - + - quantifier one or more
  - ^ - beginning of line
  - $ - end of line
  - [XxYy] – character class matching upper or lower case “X” or “”Y”

Useful commands

- Clear the contents of the current screen
  `$ clear`
- `$ logout`
  - The logout command logs your account out of the system (in a text mode).
  - This will end your terminal session and return to the login screen.
  - Some systems may have a file called `.logout` or `.bash_logout` in each user's home directory.
- `$ exit`
  - Exit the current shell. The `exit` command is similar to the `logout` command with the exception that it does not run the logout script located in the user's home directory.

Linux File System

- *hierarchical directory structure*: files are organized in a tree-like pattern of directories (folders), which may contain files and other directories, etc.
- Everything is a file:
  - Regular files
  - Directories: files listing a set of files
  - Symbolic links: files referring to the name of another file
- *root /: th*e first directory in the file system.
- Note: comparison with Windows,
  - Windows has a separate file system tree for each storage device (e.g. C-drive, D-drive, I-drive, …)
  - Linux has a single file system tree, regardless of how many drives or storage devices are attached to the computer.
    Storage devices are attached (or *mounted*) at various points on the tree.

ls command

Lists the files in the current directory, in alphanumeric order, except files starting with the “.” character.

- `$ ls -a` (all)
  Lists all the files (including .* files)
- `$ ls -l` (long)
  Long listing (type, date, size, owner, permissions)
- `$ ls -t` (time)
  Lists the most recent files first
- `$ ls -S` (size)
  Lists the biggest files first
- `$ ls -r` (reverse)
  Reverses the sort order
- `$ ls -ltr` (options can be combined)
  Long listing, most recent files at the end

ls command

- `$ ls *txt`
  The shell first replaces *txt by all the file and directory names ending by txt (including .txt), except those starting with ., and then executes the ls command line.
- `$ ls -d .*`
  Lists all the files and directories starting with .
  -d tells ls not to display the contents of directories.
  `$ ls -d */`
- `$ ls ?.log`
  Lists all the files which names start by 1 character and end by .log
- http://www.thegeekstuff.com/2009/07/linux-ls-command-examples/

ls command

```
vsc30051@login1:~/matlab/test> ls -al
total 1156
drwxr-xr-x 9 vsc30051 vsc30051   8192 2011-03-29 16:10 .
drwxr-xr-x 9 vsc30051 vsc30051   4096 2011-06-20 12:08 ..
drwxr-xr-x 3 vsc30051 vsc30051   4096 2010-01-08 10:54 courseAndreas
-rwxr--r-- 1 vsc30051 vsc30051 149326 2011-03-10 15:14 fibonacci
-rw-r--r-- 1 vsc30051 vsc30051    681 2011-03-10 15:09 fibonacci.m
-rwxr--r-- 1 vsc30051 vsc30051 149068 2011-03-10 12:16 ftest
-rw-r--r-- 1 vsc30051 vsc30051    443 2011-03-10 12:17 ftest_compiled.sh
-rw-r--r-- 1 vsc30051 vsc30051    442 2011-03-10 11:28 ftest_compiled.sh~
-rw-r--r-- 1 vsc30051 vsc30051     70 2011-03-10 12:15 ftest.m
```

- In Linux, file is defined as simply the thing that deals with a sequence of bytes
- Hence everything are files: an ordinary file is a file; a directory is also file; a network card, a hard disk, any device are also files since they deal with a sequence of bytes

Moving around

- **Symbolic (soft) link**
  - Not a real file, just a link to another file
  - Allows giving another name to a file without actually duplicating it – hence saves memory space

- **Special file (device)**
  - Each hardware device, e.g. keyboard, hard disk, CD-ROM, etc. is associated with at least one file
  - Usually store in /dev directory
  - Applications can read and write any devices by reading and writing their associate file – hence the access method is known as device independent
  - Divide into two types: character special files, e.g. keyboard, and block special files, e.g. disk

Moving around

- Display the current/working directory
  - `$ pwd`
  - **P**rint **W**orking **D**irectory
  - displays your current location within the file system.
- Change (navigate) directories.
  - `$ cd dir_name`
  - **C**hange **D**irectories
  - changes the position to the specific directory
- You can specify directory names in two ways:
  - Absolute pathname (starts from the root of the tree)

    ```
    $ cd /u/home/hpc/test/bin
    ```
  - Relative pathname (relative to your current directory)

    ```
    $ cd
    $ cd .
    $ cd ..
    $ cd test/bin
    ```

Special directories

- .
  - The current directory.
  - Useful for commands taking a directory argument.
  - Useful to run commands in the current directory
  - ./readme.txt and readme.txt are equivalent.
- ..
  - The parent (enclosing) directory. Always belongs to the . Directory
  - Typical usage:
    cd ..
- ~
  - Shells just substitute it by the home directory of the current user.
- -
  - cd – jump back to the previous directory

File paths

- A path is a sequence of nested directories with a file or directory at the end, separated by the / character
- **Relative path**: `documents/fun/file1`
  Relative to the current directory
- To work with relative paths they need to have connection on the linux tree
- **Absolute path**: `/home/user/leuven/file2`
- `/` : root directory.
  Start of absolute paths for all files on the system
  Not a superuser's home directory (-> /root)

File names

File name features:

- Case sensitive
- No obvious length limit (255 characters)
- Can contain any character (including whitespace, except /).
- File name extensions not needed and not interpreted. Just used for user convenience. (File types stored in the file)
- a hidden file is any file that begins with a “.” (not seen with the bare *ls*
- File name examples:

```
README          .bashrc             index.htm
index.html          index.html.old
```

Auto-Completion

- Have the shell automatically complete commands or file paths.
- Activated using the **<TAB>** key on most systems
- examples
  - `$ whe<TAB>`
  - `$ whereis`
  - `$ ls -l /etc/en<TAB>`
  - `$ ls -l /etc/environment`
- When more than one match is found, the shell will display all matching results (use **<TAB>** twice)
  - `$ ls -l /etc/host<TAB>`

Command history: Arrow Up

- Previously executed commands can be recalled by using the **Up Arrow** key on the keyboard.
- Most Linux distributions remember the last five hundred commands by default.
- Display commands that have recently been executed
  - The `history` command displays a user's command line history.
  - You can execute a previous command using `![NUM]` where NUM is the line number in history you want to recall.

Globbing: use wildcard

Wildcard	Function
*	Matches 0 or more characters
?	Matches 1 character
[abc]	Matches one of the characters listed
[a-c]	Matches one character in the range
[!abc]	Matches any character not listed
[!a-c]	Matches any character not listed in the range
{tacos,nachos}	Matches one word in the list

```
$ ls -l /etc/host*
$ ls -l /etc/hosts.{allow,deny}
$ ls -l /etc/hosts.[!a]*
$ ls -l /etc/host?
```

Pipes

- Pipes (also referred to as pipelines) can be used to direct the output of one command to the input of another.
  The Shell arranges it so that the standard output of one command is fed to another command
- use the **|** key on the keyboard
- `$ ls -l | less`
  (compare with `$ ls -l > less`)

Standard output

- All the commands outputting text on your terminal do it by writing to the **standard output**.
  This is where the normal output from a program goes.
- Standard output can be written (**redirected**) to a file using the > symbol

```
$ ls > /tmp/list.txt
```

```
$ date > date.txt
```

- Standard output can be **appended** to an existing file using the >> symbol

```
$ ls >> /tmp/list.txt
```

File Manipulation

- For all file manipulation commands relative of absolute paths can be used (or just files in the current directory when no extra path specified)

- Most of the commands are intuitive – shortcuts of English names

Directories

- Create directories
  - The `mkdir` command is used to create directories
  - `$ mkdir dir1 dir2 dir3`
- Remove directories
  - The `rmdir` command removes directories
  - `$ rmdir dir1`
  - `rmdir` will only remove empty directories.
    To remove a non-empty directory, use
    `$ rm -r [DIRECTORY]` instead.

Copy a file

- The `cp` command copies files and directories
- The default behavior will overwrite any existing file(s). The **-i** option overrides this behavior and prompts the user before overwriting the destination file.
- **syntax: `cp [OPTIONS] [SOURCE] [DESTINATION]`**
  - `$ cp <source_file> <target_file>`
    Copies the source file to the target.
  - `$ cp file1 file2 file3 ... dir`
    Copies the files to the target directory (last argument).
  - `$ cp -i` (interactive)
    Asks for user confirmation if the target file already exists
  - `$ cp -r <source_dir> <target_dir>` (recursive)
    Copies the whole directory.
  - `$ cp -v` (verbose)
    Displays what has been copied

Move or rename files

- Move or rename files and directories: `mv`
- The default behavior will overwrite any existing file(s).
- **syntax: `mv [OPTIONS] [SOURCE] [DESTINATION]`**
  - `$ mv old_name new_name`
    Renames the given file or directory.
  - `$ mv -i` (interactive)
    If the new file already exits, asks for user confirm
- The `mv` command can also be used to move or rename directories
  - `$ mv NewFiles/ OldFiles/`
  - -r option is not necessary 

Remove files

- The rm command removes files.
- `$ rm file1 file2 file3 ...`
  Removes the given files.
- `$ rm -i` (interactive)
  Always ask for user confirmation.
- `$ rm -r dir1 dir2 dir3` (recursive)
  Removes the given directories with all their contents.

- Tip:
  Whenever you use wildcards with `rm` (besides carefully checking your typing!), test the wildcard first with `ls`. This will let you see the files that will be deleted. Then press the up arrow key to recall the command and replace the `ls` with `rm`.

Create links

- Create links (shortcuts) to files or directories: `ln`
- Symbolic links are created when using the `-s` option with the `ln` command.
- Allows to jump to other files or locations on the file system
- Editing a symbolic link file is the same as editing the source file, but deleting the symbolic link does not delete the source file.
  - `$ ln -s file_v5.doc file_final.doc`
  - creates a symbolic link called file_final.doc that points to file_v5.doc
  - `$ ln -s /home/demo/dir1/dir2/dir3 /home/demo/jump2dir`
  - creates a symbolic link called jump2dir that points to a deep directory - allows for quicker access)

Text editors

- Text-only text editors
  - Often needed for sysadmins and great for power users
  - vi, vim
  - nano
- Graphical text editors
  - Fine for most needs
  - Gedit,
  - Kate, Nedit
  - Emacs, Xemacs
- http://www.thegeekstuff.com/2009/07/top-5-best-linux-text-editors/

Text editors

How to start:

- Create a file
  - `$ touch filename` (creates an empty file)
  - start editing (non)-existing file:

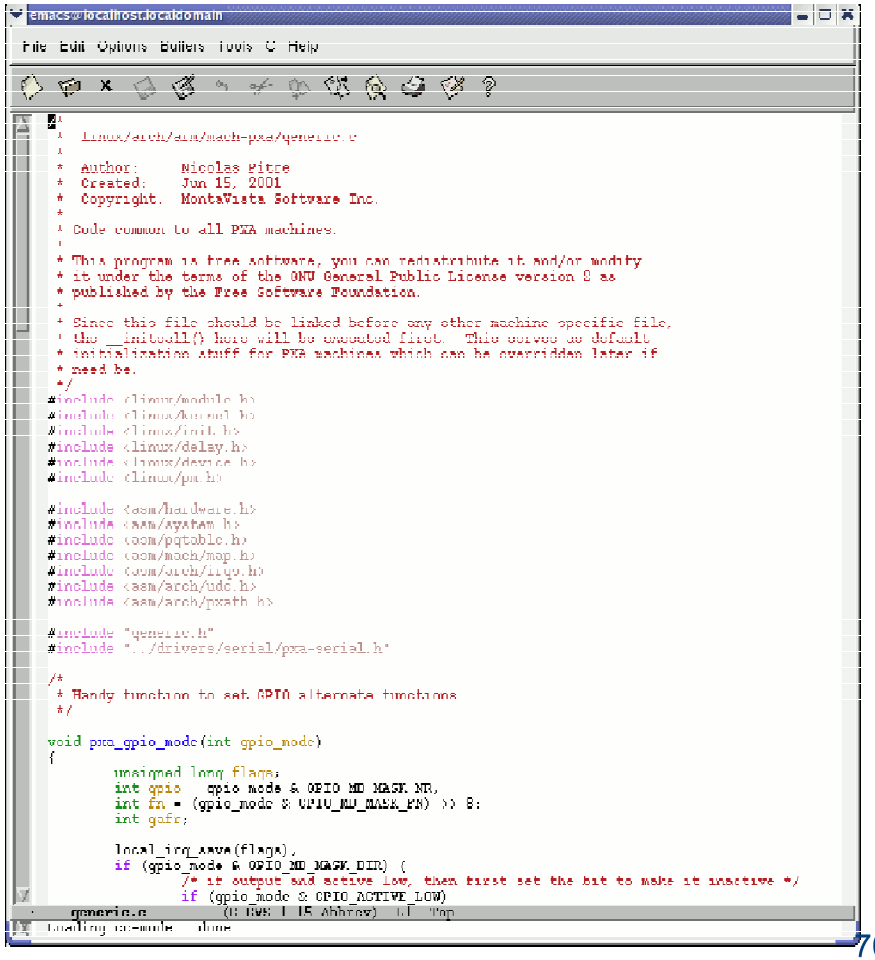
```
$ vi filename
$ nano filename
$ gedit filename
```

vi

- Text-mode text editor available in all Linux systems.
- Created before computers with mice appeared.
- Difficult to learn for beginners used to graphical text editors.
- Very productive for power users.
- Check the web for tutorials:
  - https://upload.wikimedia.org/wikipedia/commons/d/d2/Learning_the_vi_Editor.pdf
  - ftp://ftp.vim.org/pub/vim/doc/book/vimbook-OPL.pdf

Emacs

- Extremely powerful text editor features
- Great for power users
- Non standard shortcuts
- Much more than a text editor
  (games, e-mail, shell, browser).
- Some power commands have to be learnt.



Nano


u0076109@hpc-p-svcs-10:~
GNU nano 2.2.6 File: testfile
Exit=Ctrl+X
[New File]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Displaying file contents

Several ways of displaying the contents of files.

- `$ cat file1`
  displays the contents of the given file.
- `$ cat file1 file2 file3 ...` (concatenate)
  Concatenates and outputs the contents of the given files.
- `$ more file1`
  Display the output of a command or text file one page at a time.
- Can also jump to the first occurrence of a keyword (/ command).

Displaying file contents

- `$ less file1`

  Does more than more.
  Doesn't read the whole file before starting.
  Supports backward movement in the file (? command).
  Press `q` to exit

- `$ display file1`

  Displays graphical file (simple image)

The head and tail commands

- `$ head [-<n>] <file>`
  Displays the first <n> lines (or 10 by default) of the given file.
  Doesn't have to open the whole file to do this!
- `$ tail [-<n>] <file>`
  Displays the last <n> lines (or 10 by default) of the given file.
  No need to load the whole file in RAM! Very useful for huge files.
- `$ tail -f <file>` (follow)
  Displays the last 10 lines of the given file and continues to display new lines when they are appended to the file.
  Very useful to follow the changes in a log file, for example.

The grep command

- `$ grep <pattern> <files>`
  Scans the given files and displays the lines which match the given pattern.
- `$ grep error *.log`
  Displays all the lines containing error in the *.log files
- `$ grep -i error *.log`
  Same, but case insensitive
- `$ grep -ri error .`
  Same, but recursively in all the files in the current directory and its subdirectories
- `$ grep -v info *.log`
  Outputs all the lines in the files except those containing info.
- http://www.thegeekstuff.com/2009/03/15-practical-unix-grep-command-examples/

wget

- Instead of downloading files from your browser, just copy and paste their URL and download them with wget
- main features
  - http and ftp support
  - Can resume interrupted downloads
  - Can download entire sites or at least check for bad links
  - Very useful in scripts or when no graphics are available (system administration, embedded systems)

- `$ wget -c http://microsoft.com/customers/dogs/winxp4dogs.zip`

Continues an interrupted download.

- `$ wget -r -np http://www.xml.com/ldd/chapter/book/`

Recursively downloads an on-line book for off-line access.
-np: "no-parent". Only follows links in the current directory.

time

- time a helpful command for doing simple benchmarking
- run your scripts along with **time** command, and compare their's execution time.
- Example:

```
$ time ls
```

real 0m2.304s (actual elapsed time)
user 0m0.449s (CPU time running program code)
sys 0m0.106s (CPU time running system calls)

real = user + sys + waiting
waiting = I/O waiting time + idle time (running other tasks)

information about users

- `$ who`
  Lists all the users logged on the system.
- `$ whoami`
  Tells what user I am logged as.
- `$ groups`
  Tells which groups I belong to.

More commands

- `$ sleep 60`
  Waits for 60 seconds
  (doesn't consume system resources).
- `$ wc report.txt`
  word count
  Counts the number of lines, words and characters in a file or in standard input.
- `$ date`
  Returns the current date. Useful in scripts to record when commands started or completed.
- Before you run a command, `which` tells you where it is found
  `$ which ls`

Measuring disk usage

- `$ du -h <file>`

  -h: returns size on disk of the given file, in human readable format: K (kilobytes), M (megabytes) or G (gigabytes),
  Without -h, du returns the raw number of disk blocks used by the file (hard to read).
  Note that the -h option only exists in GNU du.

- `$ du -sh <dir>`

  -s: returns the sum of disk usage of all the files in the given directory.

Measuring disk space

- `$ df -h <dir>`

  Returns disk usage and free space for the filesystem containing the given directory. Similarly, the -h option only exists in GNU df.

- Example:

```
$ df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda5             9.2G  7.1G  1.8G  81% /
```

- `$ df -h`

  Returns disk space information for all filesystems available in the system. When errors happen, useful to look for full filesystems.

Comparing files and directories

- `$ diff file1 file2`
  Reports the differences between 2 files, or nothing if the files are identical.
- `$ diff -r dir1/ dir2/`
  Reports all the differences between files with the same name in the 2 directories.
- These differences can be saved in a file using the redirection, and then later re-applied.
- https://linuxacademy.com/blog/linux/introduction-using-diff-and-patch/

Other commands

- `cut` – cuts columns of text from a file
- `echo` – displays a line of text
- `paste` – will paste columns of text into a file
- `sort` – sorts a file of lines alfabetically
- `tr` – translates between characters (e.g. tr a-z A-Z)
- `rpm -q` – query about installed software
- `whereis` - locate the binary, source, and manual page files for a command
- `which` - shows the full path of (shell) commands
- `bc` – command line calculator (scale=2 to see 2 digits)

File Archiving: tar

- Saves and restores multiple files to/from a single file. Directories followed recursively.
- Format:
  - `$ tar [options] [options_values] [files]`

  - c – create a new archive
  - v – verbosely list files which are processed.
  - f – following is the archive file name
  - z – filter the archive through gzip (compress)
  - x – extract files from archive
  - j - filter the archive through bzip (compress)

File Archiving: tar

- Examples:
  - `$ tar -cvf [FILE] [ITEM]` Backup the specified item(s)

    `$ tar -cvf /tmp/backup.tar ~/data ~/test`
  - `$ tar -czvf [FILE] [ITEM]` Compress the archive to save space
  - `$ tar -xvf [FILE] [ITEM]` Restore the specified item(s) **`$tar -xvf backup.tar`**
  - `$ tar -tf [FILE]` List all files in the specified archive

    e.g. `$ tar -tf backup.tar`
- http://www.thegeekstuff.com/2010/04/unix-tar-command-examples/

File Compression: gzip

- Compressing files: `gzip filename` or `bzip2 filename`
  - `$ gzip backup.tar`
  - `$ bzip2 backup.tar`

  The resulted file is backup.tar.gz/ backup.tar.bz2
- Uncompressing files: `gzip -d filename.gz` or `bzip2 -d filename.bz2`
  - `$ gzip -d backup.tar.gz`
  - `$ bzip2 -d backup.tar.bz2`

  The uncompressed file is backup.tar

File access rights

Linux File Access Privilege

- Linux is a multiuser system, the files of all users are stored in a single file structure
- Mechanism is required to restrict one user to access the files of another user, if he is not supposed to
- User can impose access permission to each file to restrict its access
- The term “access permission” refers to
  - read permission
  - write permission
  - execute permission

File access rights

3 types of **access rights**

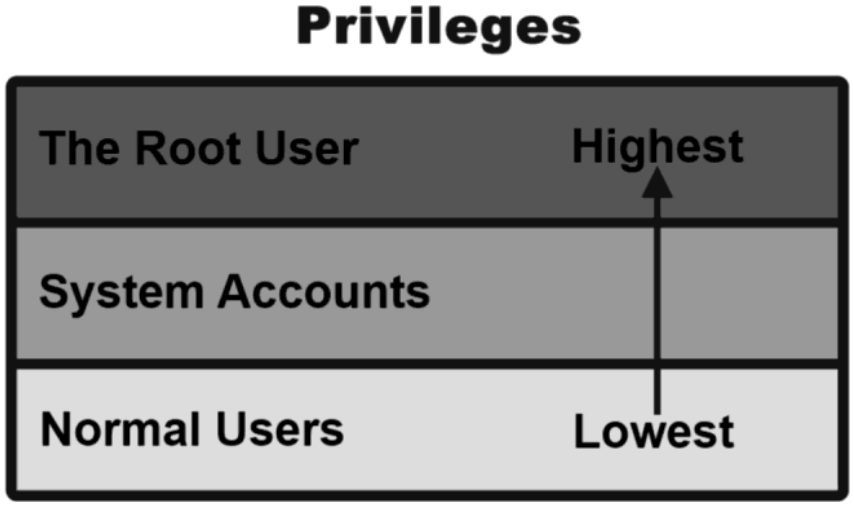
- Read access (r)
  - reading, opening, viewing, and copying the file is allowed
- Write access (w)
  - writing, changing, deleting, and saving the file is allowed
- Execute rights (x)
  - executing and invoking the file is allowed. This is required for directories to allow searching and access.

Use `ls -l` to check file access rights

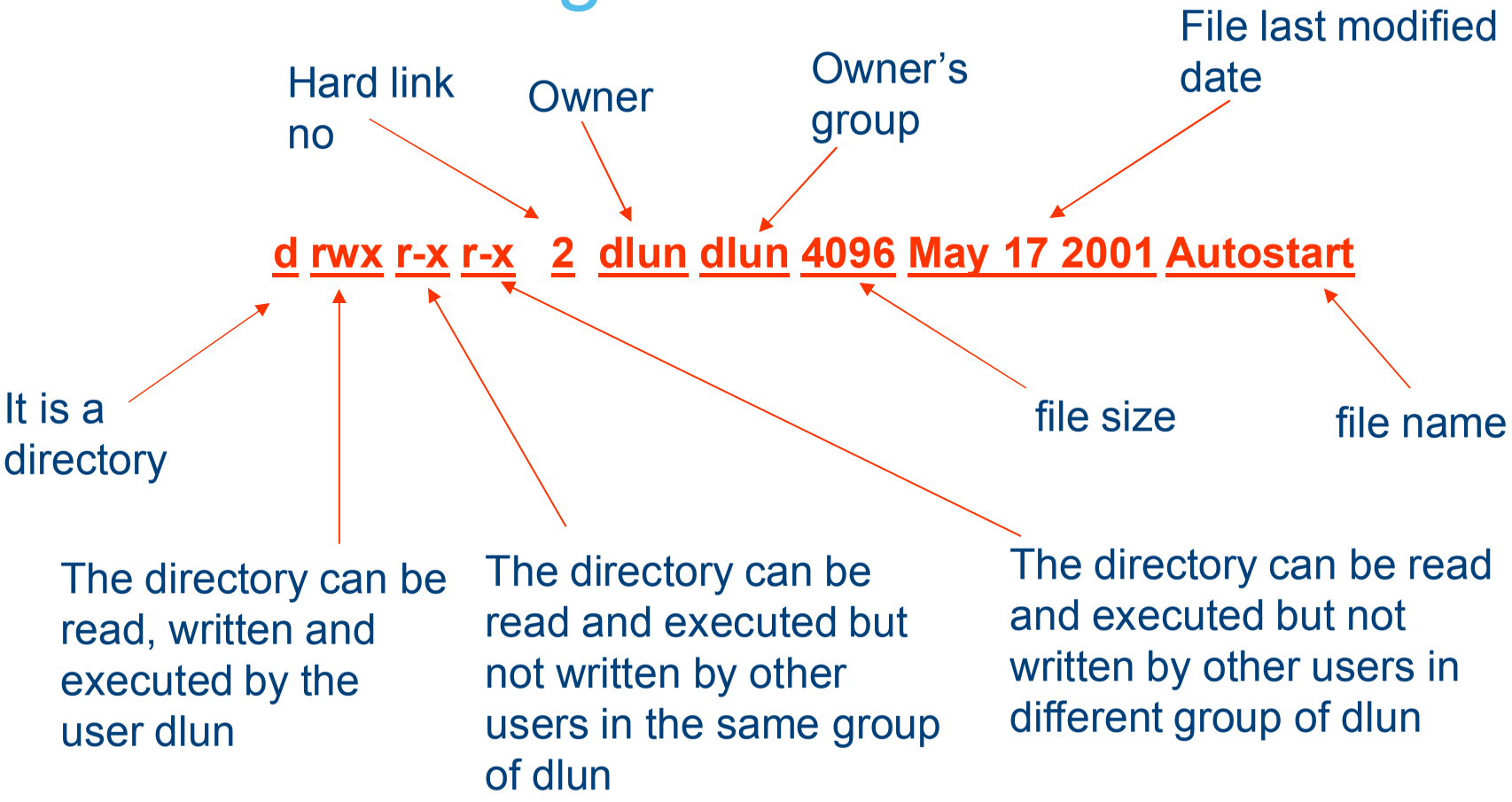
File access rights

3 types of **access levels**

- User (u): for the owner of the file
- Group (g): each file also has a “group” attribute, corresponding to a given list of users
- Others (o): for all other users



File access rights



The group of a user is assigned by the administrator when a user is added to the system

File access rights

- Access permission can also be assigned to a directory
- Directory is also a file that contains the attributes of the files inside it
- If read permission is not given to a directory
  - cannot show the structure of this directory
  - e.g. cannot use ls
- If write permission is not given to a directory
  - cannot modify anything of the directory structure
  - e.g. cannot copy a file into this directory since it will modify the directory structure by adding one more file
- If execute permission is not given to a directory
  - nearly nothing can be done with this directory, even cd

Access rights examples

- -rw-r--r--
  Readable and writable for file owner, only readable for others
- -rw-r-----
  Readable and writable for file owner, only readable for users belonging to the file group.
- drwx------
  Directory only accessible by its owner
- -------r-x
  File executable by others but neither by your friends nor by yourself. Nice protections for a trap...

Access right constraints

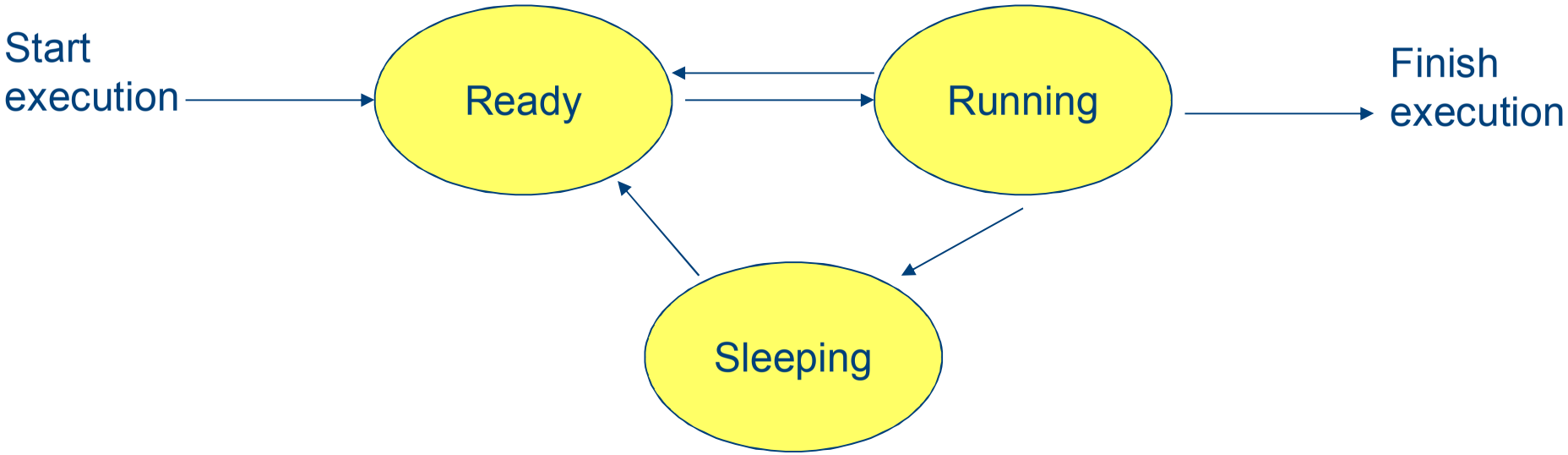
- x is sufficient to execute binaries
  Both x and r and required for shell scripts.
- Both r and x permissions needed in practice for directories:
  r to list the contents, x to access the contents.
- You can't rename, remove, copy files in a directory if you don't have w access to this directory.
- If you have w access to a directory, you can remove a file even if you don't have write access to this file (remember that a directory is just a file describing a list of files). This even lets you modify (remove + recreate) a file even without w access to it.

Process

- “Everything in Unix is a file. Everything in Unix that is not a file is a process”
- Processes
  - Instances of a running programs
  - Several instances of the same program can run at the same time
  - Processes are assigned a unique identifier which is used to monitor and control the process (PID)

Process

- A program that is claimed to be executing is called a process
- For a multitasking system, a process has at least the following three states:



Process

- Ready state
  - All processes that are ready to execute but without the CPU are at the ready state
  - If there is only 1 CPU in the system, all processes except one are at the ready state
- Running state
  - The process that actually possesses the CPU is at the running state
  - If there is only 1 CPU in the system, at most there is only one process is at the running state
- Sleeping state
  - The process that is waiting for other resources, e.g. I/O, is at the sleeping state

ps

- Display running processes
  (cfr. Windows Task Manager ctrl-shift-esc)
- `$ ps` Display the current user's processes
- `$ ps -e` Display all processes running on the system
- `$ ps -ef` Display detailed information about running processes
- `$ ps -u [USER]` Display processes owned by the specified user
- `$ ps a` Display extra info (running state)

kill

- Sends an abort signal to the given processes. Lets processes save data and exit by themselves. Should be used first.
- `$ kill <pid>`
  Example:
  `$ kill 3039 3134 3190 3416`
- `$ kill -9 <pid>`
  Sends an immediate termination signal. The system itself terminates the processes. Useful when a process is really stuck.

killall

- The `killall` command terminates all processes that match the specified name
- `$ killall [-<signal>] <command>`
  Example:
  `$ killall bash`

xkill

`xkill`

Lets you kill a graphical application by clicking on it!
Very quick! Convenient when you don't know the application command name.

&

- `&` is a command line operator that instructs the shell to start the specified program in the background.
- This allows you to have more than one program running at the same time without having to start multiple terminal sessions.
- Starting a process in background: add & at the end of your line:
  `$ gedit &`
  check with `ps`