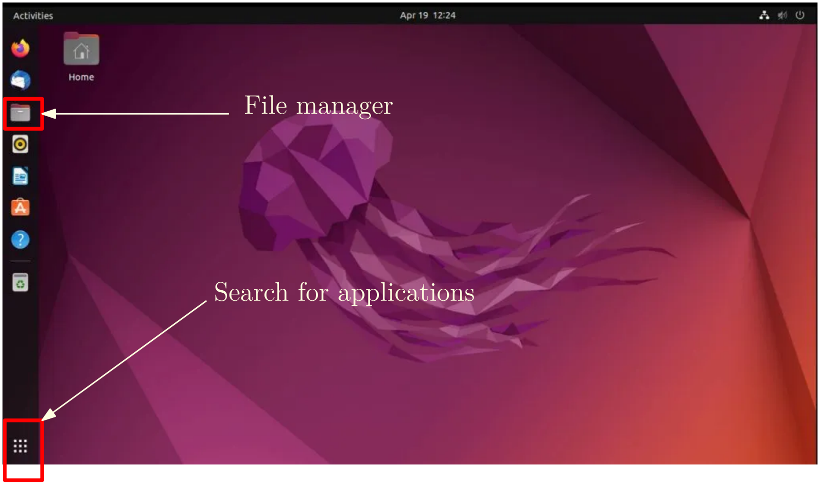
Introduction to UNIX-like systems

Computing Laboratory

`http://www.isical.ac.in/~dfslab`

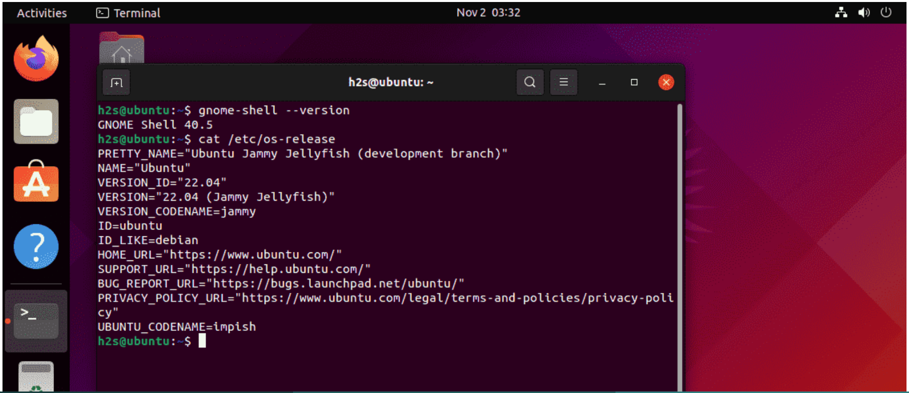
Outline

Your desktop



What you will need

- Code editor: Visual Studio Code (if you do not already have a preferred editor)
  - search for "`linux vscode setup for c programming`"
- Text editor (gedit): for quick edits
- Terminal (image source: `https://linux.how2shout.com/`)

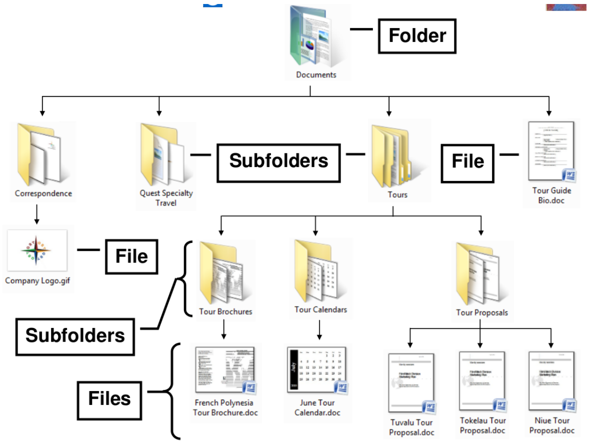


```
h2s@ubuntu:~$ gnome-shell --version
GNOME Shell 40.5
h2s@ubuntu:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu Jammy Jellyfish (development branch)"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04 (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-poli
cy"
UBUNTU_CODENAME=impish
h2s@ubuntu:~$
```

Outline

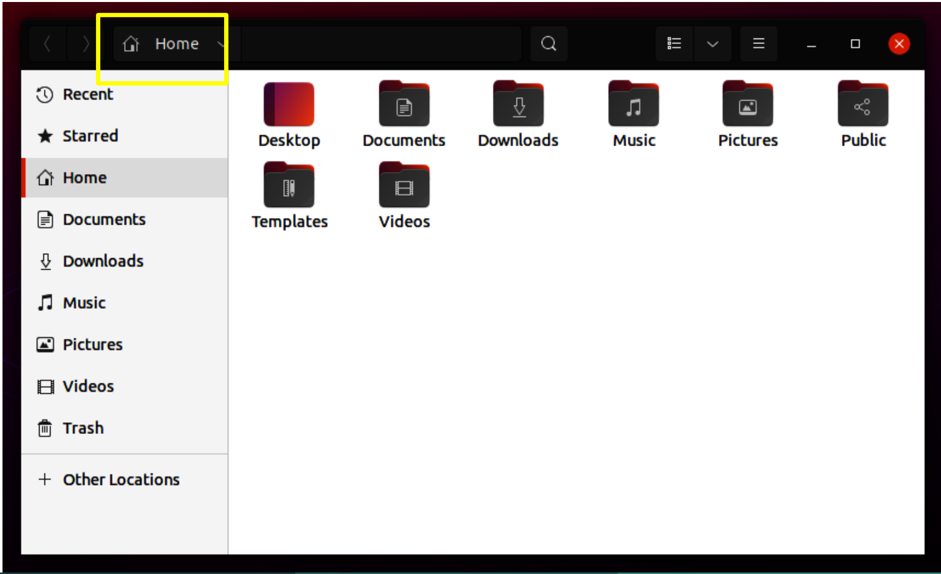
File system structure

Files are organised in a hierarchical structure of folders, sub-folders, and files.

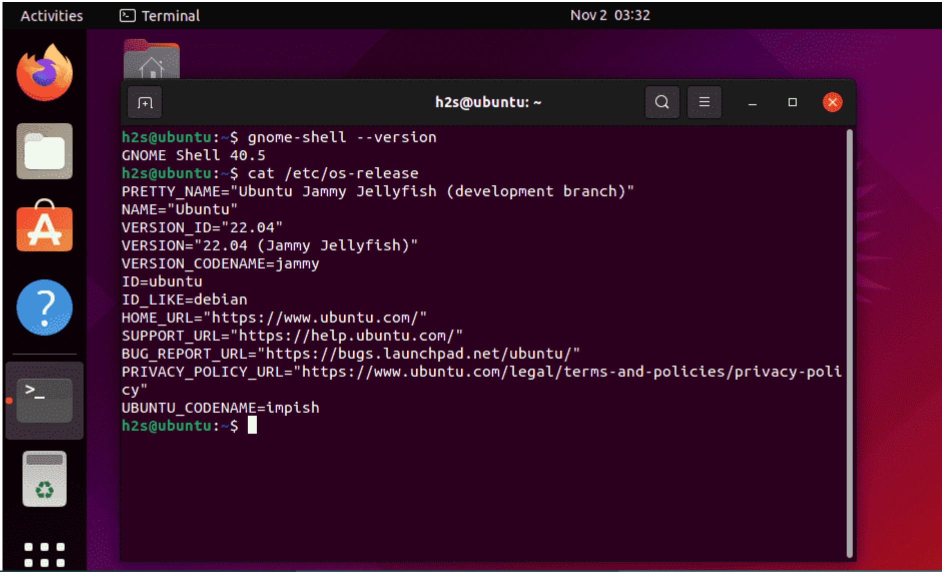


Courtesy: `https://www.slideshare.net/okmomwalking/windows-7-unit-b-ppt`

File system hierarchy



File system hierarchy



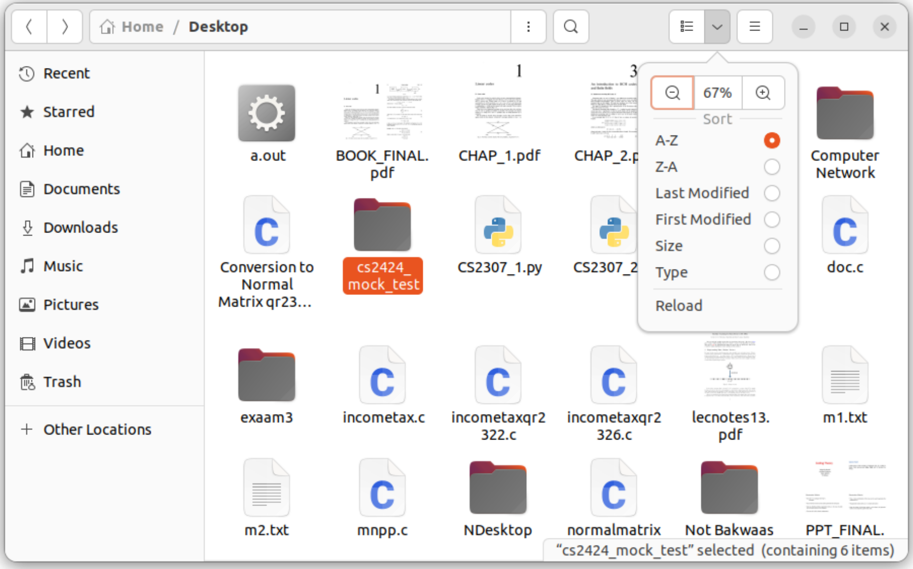
```
h2s@ubuntu:~$ gnome-shell --version
GNOME Shell 40.5
h2s@ubuntu:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu Jammy Jellyfish (development branch)"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04 (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-poli
cy"
UBUNTU_CODENAME=impish
h2s@ubuntu:~$
```

File system structure: terminology

- Folders ≡ *directories*
- Top of the hierarchy: *root directory* (/)
- Default starting location: *home directory* (~)
- Location of a file or directory: specified by *path*
- Current location in terminal or file browser: *current working directory*
- Paths: *absolute* or *relative*
  - absolute path: from root (starts with /)
    Example: `/usr/bin/firefox`, `/tmp`, `/user1/student`
  - relative path: from current working directory (does not start with /)
    Example: `clab/assignment1/hello.c`

Note the difference between (forward) slash (/, used in Unix-like systems) and backslash (\, used in Windows-like systems) !

Graphical file manager



Managing files from the terminal

Navigating the file system

- `cd` : change directory
  Example:
  ```
  cd /home/student/Desktop
  cd clab/day1/
  cd
  ```
  $\longleftarrow$ go to home directory
- `pwd` : print current working directory

Managing files from the terminal

Navigating the file system

- `cd` : change directory
  Example:
  ```
  cd /home/student/Desktop
  cd clab/day1/
  cd
  ```
  $\longleftarrow$ go to home directory
- `pwd` : print current working directory

Special directory names

- $\sim$: home directory
  Example: `cd ~/Desktop`
- `.` (dot) : current working directory
  Example: `./program1`
- `..` (dot dot) : parent directory (one level up)
  Example: `cd ..`, `cd ../assignment2`, `cd ../..`

Managing files from the terminal

Listing files

- `ls` : view list of files in current directory
- `ls <path>` : view list of files in specified path
- `ls -l` : view detailed list of files
- `ls -lt` : view detailed list of files sorted by modification time
- `ls -ltr` : view detailed list of files sorted by modification time *in reverse order*

Managing files from the terminal

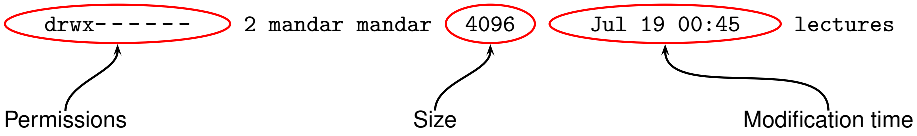
Listing files

- `ls` : view list of files in current directory
- `ls <path>` : view list of files in specified path
- `ls -l` : view detailed list of files
- `ls -lt` : view detailed list of files sorted by modification time
- `ls -ltr` : view detailed list of files sorted by modification time *in reverse order*

Example:

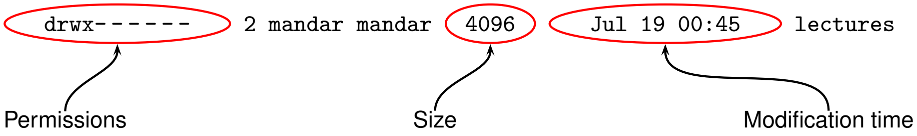
```
$ /bin/ls -l
total 68
drwx------ 2 mandar mandar  4096 Jul 19 00:45 assignments
drwx------ 2 mandar mandar  4096 Jul 22  2016 exams
-rw-r--r-- 1 mandar mandar 13521 Jul 19 00:41 index.html
drwx------ 2 mandar mandar  4096 Jul 19 00:45 lectures
```

Essential commands: permissions



```
drwx------ 2 mandar mandar 4096 Jul 19 00:45 lectures
```

Essential commands: permissions



```
drwx------ 2 mandar mandar 4096 Jul 19 00:45 lectures
```

Permissions:

- 9 possible permissions:
  { read, write, execute } $\times$ { user (owner), group, other (everyone else) }
- 9 bits (1 $\equiv$ permission granted)

ur	uw	ux	gr	gw	gx	or	ow	ox

- `chmod`: changing permissions

  Example:

```
chmod g+wx <path>
chmod og-wx <path>
chmod 644 <path>
chmod 700 <path>
```

$\longleftarrow 644 \equiv 110\ 100\ 100 \equiv$ `rw-r--r--`

$\longleftarrow 700 \equiv 111\ 000\ 000 \equiv$ `rwx------`

Essential commands: directories

- `mkdir` : create a directory
  Examples:
  `mkdir clab`
  `mkdir clab/assignment1`

  Create directories as appropriate.

  OR
  `mkdir -p clab/assignment1 clab/assignment2`
- `rmdir` : remove an (empty) directory
  Example: `rmdir assignment2`, `rmdir clab/programs`

Outline

Compiling and running your program

- Compiling

  command options / flags other arguments

  ```
  gcc -g -Wall -o prog1 prog1.c
  ```

  OR

  ```
  gcc -g -Wall prog1.c
  ```

- Running

  `./prog1` OR `./a.out`

  OR

  `./prog1 input.txt output.txt`

  etc.

Essential commands: files

- `cp` : copy a file
  Example:
  ```
  cp program1.c program2.c
  cp -i source-file target-file
  cp -i source-file target-directory
  ```

Essential commands: files

- `cp` : copy a file
  Example:
  ```
  cp program1.c program2.c
  cp -i source-file target-file
  cp -i source-file target-directory
  ```
- `mv` : rename (move) a file
  Example:
  ```
  mv program1.c program2.c
  mv -i source-file target-file
  mv -i source-file target-directory
  ```

Essential commands: files

- `cp` : copy a file
  Example:
  ```
  cp program1.c program2.c
  cp -i source-file target-file
  cp -i source-file target-directory
  ```
- `mv` : rename (move) a file
  Example:
  ```
  mv program1.c program2.c
  mv -i source-file target-file
  mv -i source-file target-directory
  ```

`-i` ≡ interactive
(asks for confirmation)

- `rm` : remove (delete) a file
  Example:
  ```
  rm program1.c
  rm -i file1 file2.c *.bak
  ```
  `rm -r some-directory` (remove directory and everything inside it)

Viewers / pagers

- Useful for quickly viewing a file (not editing)
- Use `less`

  Example: `less cs19xx-day0-prog1.c`

  - space: move forward one page
  - backspace or b: move backward one page
  - q : exit the pager
  - / : search for a string in the file
  - run `man less` for more information

Input/output redirection

- Input/output
  - involves a file or a terminal
  - requires a *file pointer*
- *stdin* $\equiv$ file pointer corresponding to reading input from keyboard
- *stdout* $\equiv$ file pointer corresponding to printing output to terminal
- Taking stdin from a file: `./prog1 < input.txt`
- Printing stdout to a file: `./prog1 > input.txt`
- Taking stdin / printing stdout from / to a program: use the vertical bar / pipe character (|)

```
./prog1 | less
cat input.txt | ./prog1
```

- Input/output from/to file / program may be combined

```
./prog1 < input.txt > output.txt
```

Other commands

- `man`
  Example: `man ls`, `man cp`, `man rm`

Other commands

- `man`
  Example: `man ls`, `man cp`, `man rm`

Find out more about these on your own.

- `alias` (giving your own, easy-to-remember names to commands)
- `wc` (counting characters, words, lines)
- `sort`
- `head`, `tail` (first few / last few lines)
- `cmp`, `diff` (comparing two files)
- `ps`, `top`, `kill` (checking what programs are running)
- `find` (finding files or directories)
- `grep` (searching for patterns)
- `awk`, `sed` (programming)

Useful references / cheat-sheets

http://cli.learncodethehardway.org/bash_cheat_sheet.pdf

https://ubuntudanmark.dk/filer/fwunixref.pdf

http://www.ucs.cam.ac.uk/docs/leaflets/u5

http://mally.stanford.edu/~sr/compuGng/basic-unix.html

http://www.math.utah.edu/lab/unix/unix-commands.html