Introduction to UNIX-like systems

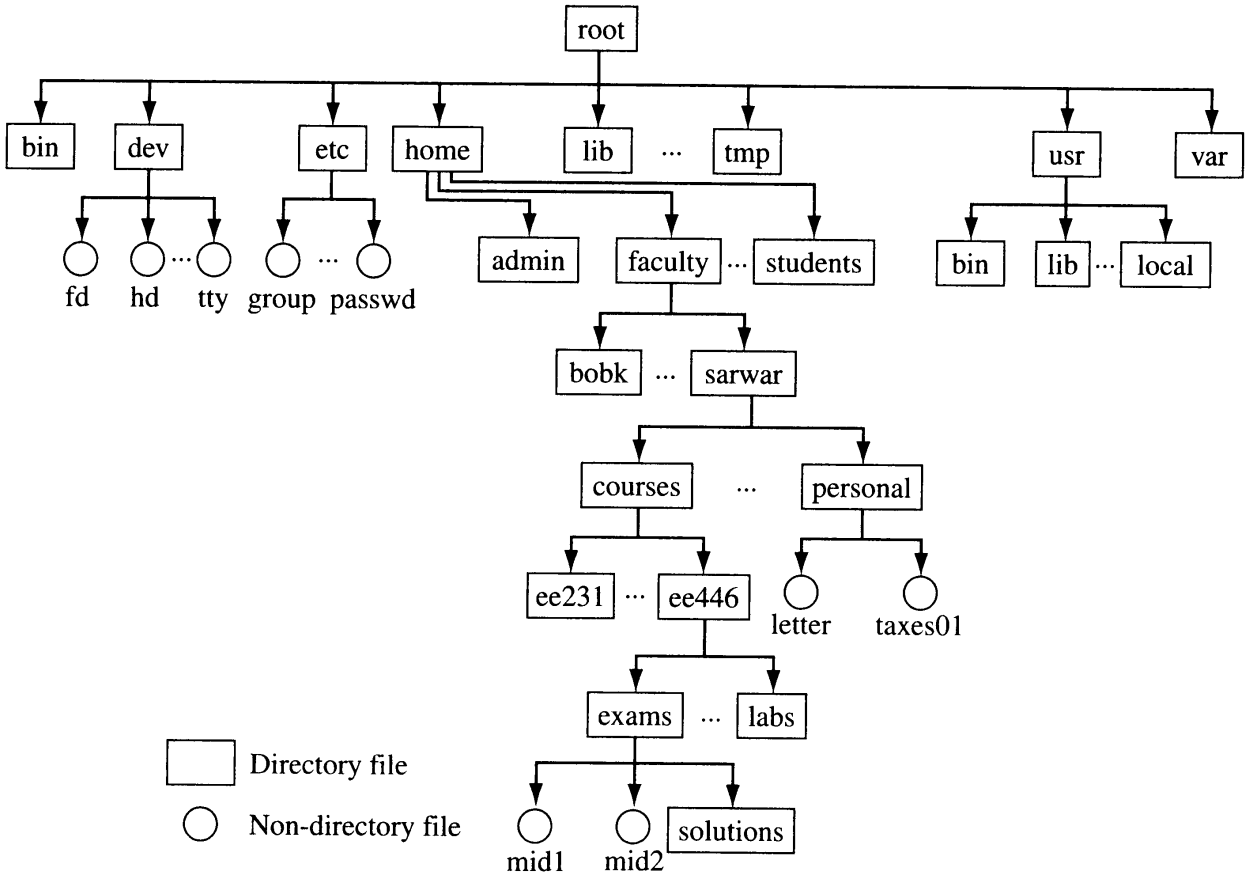
Data and File Structures Laboratory

http://www.isical.ac.in/~dfslab/2017/index.html

File system hierarchy

File system structure

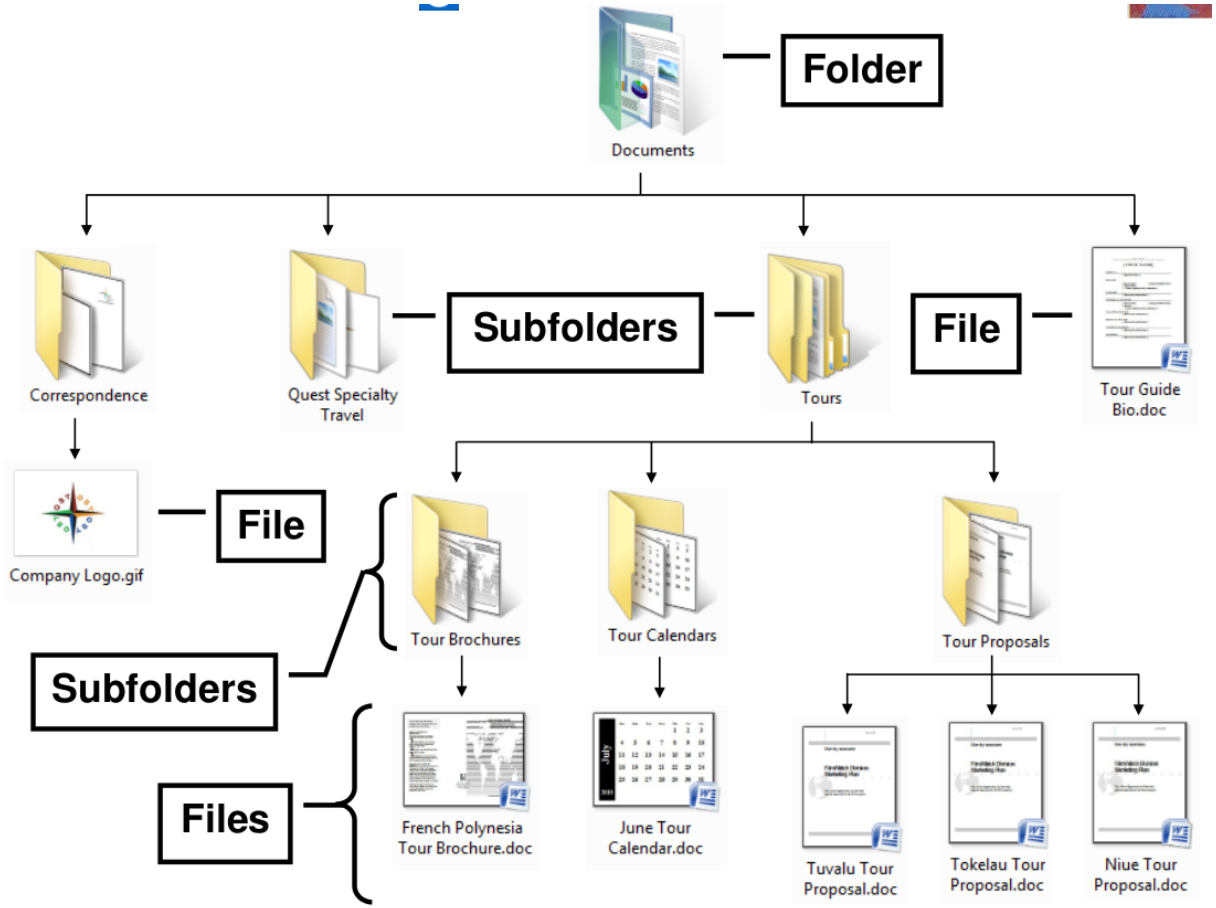
Files are organised in a hierarchical structure of folders, sub-folders, and files.



Courtesy: `http://www.cs.miami.edu/home/geoff/Courses/CSC322-11S/Content/UNIXUse/FileSystem.shtml`

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 structure: terminology

- Folders $\equiv$ *directories*
- Top of the hierarchy: *root directory* (/)
- Location of a file or directory: specified by *path*
- Current location in terminal or file browser: *current working directory*
- Normal (or default) start location: *home directory*
- Paths: *absolute* or *relative*
  - absolute path: from root
    Example: `/usr/bin/firefox`, `/tmp`, `/user1/student`
  - relative path: from current working directory
    Example: `pdslab/assignment1/hello.c`

Navigating the file system

Commands:

- `cd`: change directory
  Example:
  ```
  cd /user1/student/mtc1799
  cd pdslab/assignment1/
  cd
  ```
- `pwd`: print current working directory

Navigating the file system

Commands:

- `cd`: change directory
  Example:
  ```
  cd /user1/student/mtc1799
  cd pdslab/assignment1/
  cd
  ```
- `pwd`: print current working directory

Special directory names

- ~ : home directory
  Example: `cd ~/pdslab`
- `.` : current working directory
  Example: `./program1`
- `..` : parent directory (one level up)
  Example: `cd ..`, `cd ../assignment2`

Commands

Essential commands

- `passwd` or `yppasswd` : change your password
- `mkdir` : create a directory
  Example: `mkdir assignment2`, `mkdir pdslab/programs`
- `rmdir` : remove an (empty) directory
  Example: `rmdir assignment2`, `rmdir pdslab/programs`

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
  ```
- `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)

Essential commands: file listing

- `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

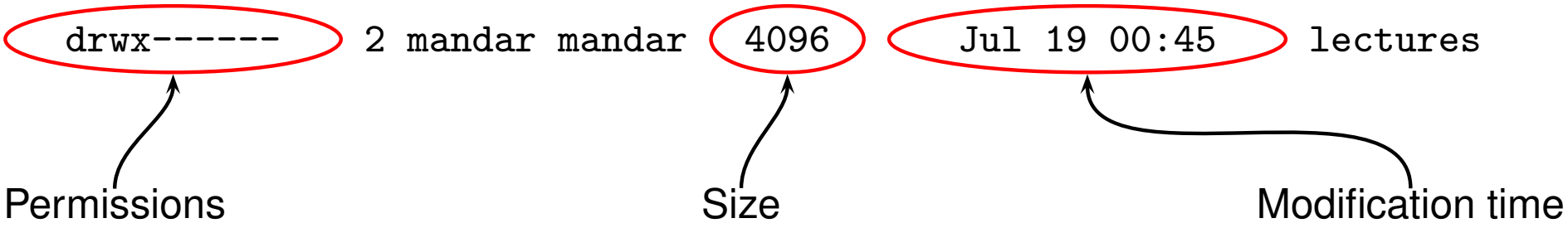
Essential commands: file listing

- `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

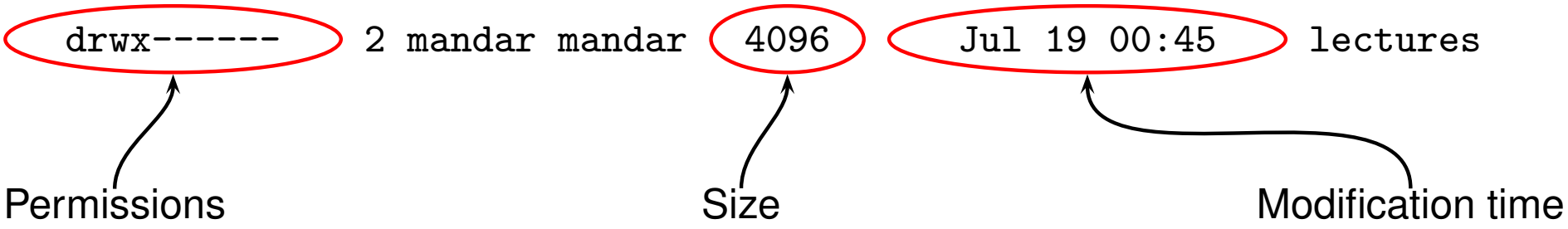
Example:

```
$ /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



Essential commands: permissions



Permissions:

- 9 possible permissions:
  { read, write, execute } × { user (owner), group, other (everyone else) }
- 9 bits (1 ≡ 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>
```

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

DFS Lab conventions

Server details

- All your work should be done on `192.168.64.35`
- To connect:

```
ssh -X mtc17xx@192.168.64.35
```

- Change password after logging in for the first time

File / directory naming conventions

- Location

```
$ cd
```

Go to your home directory.

```
$ mkdir -p pdslab/day1
```

Create a directory for today's (if you have not already done so).

```
$ cd pdslab/day1
```

Go to directory for today's class.

File / directory naming conventions

- Location

```
$ cd
```
Go to your home directory.

```
$ mkdir -p pdslab/day1
```
Create a directory for today's (if you have not already done so).

```
$ cd pdslab/day1
```
Go to directory for today's class.

- File names
  - Class work: `cs17xx-dayz-progy.c`
  - Assignments: `cs17xx-assignz-progy.c`
  - `xx` = your roll number
    `z` = day number (today is day 2)
    `y` = program number

File / directory naming conventions

At the beginning of **any** program file (class work / assignment), please write:

```
/*--------------------------------
Name:
Roll number:
Date:
Program description:
Acknowledgements:
--------------------------------*/
```

Editors

Choose any one that you like.

- `http://projects.gnome.org/gedit/`

  ```
  gedit cs17xx-day1-prog1.c &
  ```

- `http://www.nano-editor.org/`
- `http://kate-editor.org/about-kate/`
- Also, atom, emacs, geany, vim, ...

Editors

Choose any one that you like.

- `http://projects.gnome.org/gedit/`

  ```
  gedit cs17xx-day1-prog1.c &
  ```

- `http://www.nano-editor.org/`
- `http://kate-editor.org/about-kate/`
- Also, atom, emacs, geany, vim, ...

Some random opinions / guides:

- `http://lifehacker.com/five-best-text-editors-1564907215`
- `http://www.techradar.com/news/the-best-free-text-editor-2017`
- `https://www.codementor.io/mattgoldspink/best-text-editor-atom-sublime-vim-visual-studio-code-du10872i7`
- `http://blog.liveedu.tv/10-best-text-editors-programming-2016/`

Viewers / pagers

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

  Example: `less cs17xx-day1-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

Programs

1. Given two positive integers, find their greatest common divisor (gcd).
2. Given the (x, y) coordinates of the 3 vertices of a triangle, find its area.
3. Given a list of integers, find the maximum, minimum, and average.

Compiling and running

- Compiling

```
gcc -g -Wall -o prog1 cs17xx-day1-prog1.c
```

- Running

```
./prog1
```