Linux history; File system

CISC220

More silly Linux fun

```
sudo apt-get install pacman4console
pacman4console
```

A brief history of Linux

- Linux is actually the kernel of the GNU/Linux operating system (OS)
  - kernel provides an interface between hardware and software applications
- Linux is modelled on the UNIX kernel
  - UNIX was developed in the late 1970s at Bell Labs
  - became one of the first portable OSes because it was mostly written in C
  - too expensive for individual users

A brief history of Linux

- in 1983, Richard Stallman started the GNU project with the goal of creating a free, Unix-like OS
  - developed the tools needed for kernel development but was still missing an actual kernel
  - kernel was called Hurd but development was slow
- in 1991, Finnish graduate student Linux Torvalds announced on Usenet that he was developing a free operating system (as a hobby) for 80386 processors
  - used GNU software for Linux kernel development
  - thus GNU/Linux was born

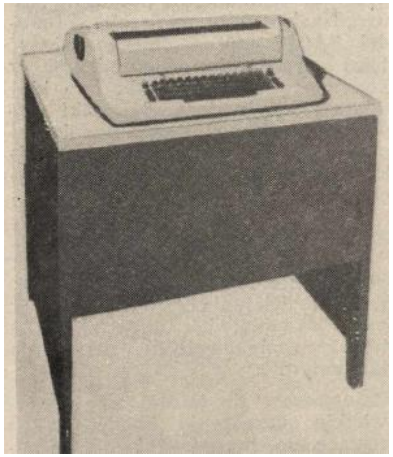
Linux today

- dominant in cloud infrastructure and supercomputing
- Android runs more than 70% of smartphones in the world
- common in embedded systems
- uncommon on the desktop (unless you include Chrome OS)

Bash

Terminals

- https://en.wikipedia.org/wiki/Computer_terminal
  - originally, an electronic or electromechanical device for data input/output



  - today, a software program called a terminal emulator
    - show

Shells

- software program that interprets text to determine meaning
  - where does the text come from? a terminal
- many different shells
  - https://en.wikipedia.org/wiki/Comparison_of_command_shells

Bash

- Bourne Again Shell
  - replaces the older UNIX Bourne shell
  - default login shell for most Linux distributions
    - was the default login shell for macOS until 2019
      - macOS now uses the zsh
- this course assumes that you are using a Bash shell

Filesystem

Filesystem

- a filesystem is the part of an operating system that controls how data is stored and retrieved
  - most often associated with physical storage media (disk, USB drive, optical media, magnetic tape, etc)
- many different filesystems
  - Windows
    - FAT, exFAT, NTFS
  - macOS
    - AFS (Apple File System)
  - Linux
    - ext, ext2, ext3, ext4, ReiserFS, XFS, …

Filesystem

- a *device* is a physical piece of hardware that can store files, e.g.,
  - disk drive, USB flash drive, CD-ROM drive

Filesystem

- the contents of a file is just bits (ones and zeros)
- imagine that your OS abstracts your disk drive as an array

Filesystem

- the contents of several files is stored in the array
  - contents of each file shown in different colors

- what information does the OS need to store so that it knows where the contents of a file can be found?
  - what other information might be useful?

Inodes

- an inode stores information about the contents of a file such as:
  - the device where the inode resides
  - file type
  - user ID (owner of the file)
  - file size
  - time when the file was created
  - time when the file was last modified
  - time when the file was last accessed
  - and more (including where to find the file contents)

Inodes

- an inode has an integer number called the *inode number* that is a unique id for the inode on a given device
  - different devices can have inodes with the same inode number so the inode number is not a globally unique id for the filesystem

Directories

- but where are the filenames?
- a directory is a file whose contents is basically a table mapping filenames to inode numbers
  - for example, the contents of my home directory might look something like:

filename	inode number
dir1	45201
dir2	45212
dir3	45229
file1.txt	43108
file2.txt	43161
file3.txt	43162
tmp	45217

Filesystem Hierarchy Standard

- https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard
- defines the *directory structure* and *directory contents* for Linux distributions
  - macOS uses a similar, but different, hierarchy
- we do not need to study this in detail but it is useful to know something about it

- there is a top-level (or uppermost) directory named / that contains all other directories
  - slash, or the root directory

To change to the root directory, use the built-in shell command **`cd`** which changes the current working directory of the shell to the specified directory:

```
cd /
```

To print the name of the current working directory use the built-in shell command **`pwd`**:

```
pwd
```

To list the contents of a directory use the **`ls`** command (ell-ess, not one-ess):

```
ls
```

Built-in commands vs commands

- there is a distinction between built-in shell commands and other commands
  - built-in commands
    - commands that are defined by the Bash shell
  - other commands
    - just programs that can be run by the Bash shell

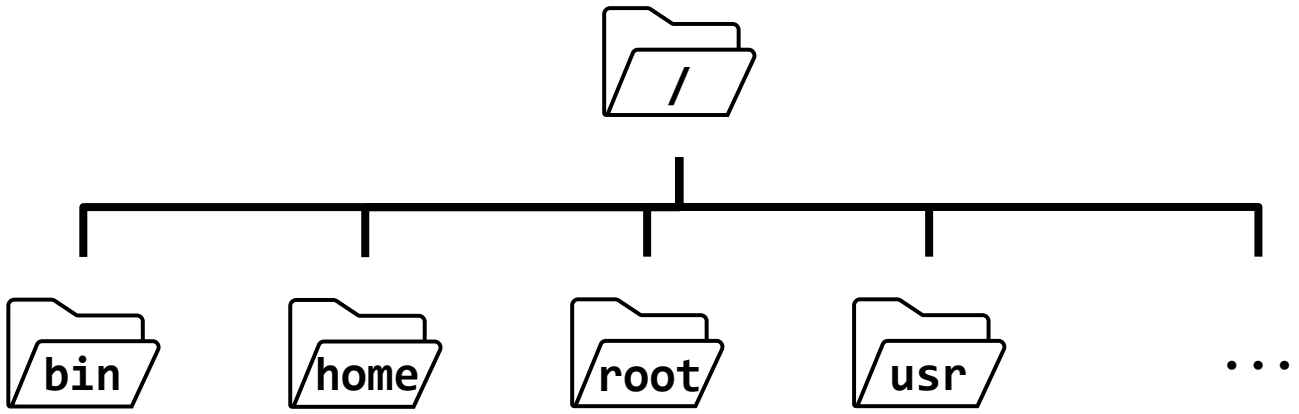
To get documentation for a built-in command, use:

```
man builtins
```

To get documentation for other commands, use **`man`** ***`cmdname`*** where ***`cmdname`*** is the name of the command:

```
man ls
```

Filesystem Hierarchy Standard



- **`bin`**: essential command binaries
- **`home`**: user home directories
- **`root`**: the root, or superuser, home directory
- **`usr`**: secondary hierarchy for read-only user data; contains majority of installed applications

To list the contents of a directory, use **`ls` *`dirname`*** where ***`dirname`*** is the name of the directory:

```
ls /bin
```

Absolute pathnames

- a pathname is the name of a file that uniquely identifies the location of a file
- an absolute pathname includes all of directory names that lead to the file
  - directories are separated by the **/** character, e.g.:

/	root directory
/bin	**bin** directory
/bin/ls	**ls** file
/home/burton/CISC220/tests/exam/exam.pdf	**exam.pdf** file

Relative pathnames

- a relative pathname includes all of directory names that lead to the file starting from the current directory
  - for example, suppose that we are in the directory **`/home/burton/CISC220`**, then the following are all relative pathnames:

`tests`	**`tests`** directory
`tests/exam`	**`exam`** directory
`tests/exam/exam.pdf`	**`examp.pdf`** file
`tests/exam/exam.docx`	**`exam.docx`** file

If the current working directory is the root directory, then we can list the contents of any other directory using a relative path:

```
ls bin
ls etc
ls etc/calendar
ls usr/local/bin
```

Any command that requires a pathname will accept a relative pathname. For example, starting from the root directory:

```
cd etc
pwd
cd calendar
pwd
ls
```

`cd` on its own will change to the current user's home directory:

```
cd
pwd
```

~, . and ..

- ~ represents the current user's home directory
  - but this can be changed
- . represents the current working directory
- .. represents the parent directory of the current working directory

```
cd ~
pwd
cd .
pwd
cd ..
pwd
cd
pwd
cd ../..
```