Files and Directories

Class 2

Administrative

- video on
- "Hello" in chat tool
- preferred first name and last name in zoom profile

- questions about anything in Monday's material?
- questions about the assignment due tomorrow?

Assignments

- you can re-submit as many times as you wish
- until the due time

- if you submit after it's due
  1. I may not see it unless you let me know
  2. it may get a late penalty

Getting Started

- essential to get started with Linux:
  1. know how to open a terminal window running BASH
  2. know how to enter a syntactically correct command on the command line
  3. understand how to create, maintain, and organize files and directories
  4. understand the importance of logging off cleanly

Getting Started

- this assumes you are connected to sand with the XFCE window manager
  - if you are new to Linux, I strongly suggest XFCE
  - if you already have Linux on your personal computer, you probably use either (GNOME) Unity or KDE as your window manager
  - if you already know how to use those, fine
  - if not, I strongly suggest XFCE which is simple and fast

Switching to XFCE

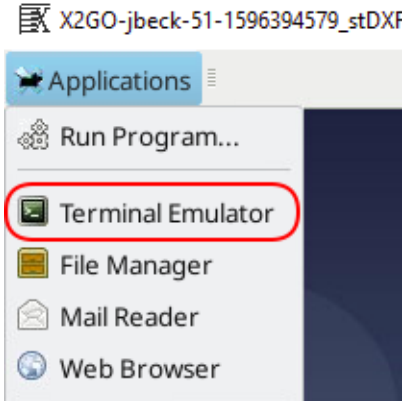
to convert your own computer's Linux desktop environment to XFCE, if you have Ubuntu or Debian:

- open a terminal window
- issue the command:
  $ sudo apt install xubuntu-desktop
- accept any dependencies and allow the installation to complete
- log out and log back in, choosing the XFCE desktop

Open a Terminal Window

- to open a terminal window, either:
  - click on the Terminal Emulator icon in the quick-start menu at the bottom center of the screen

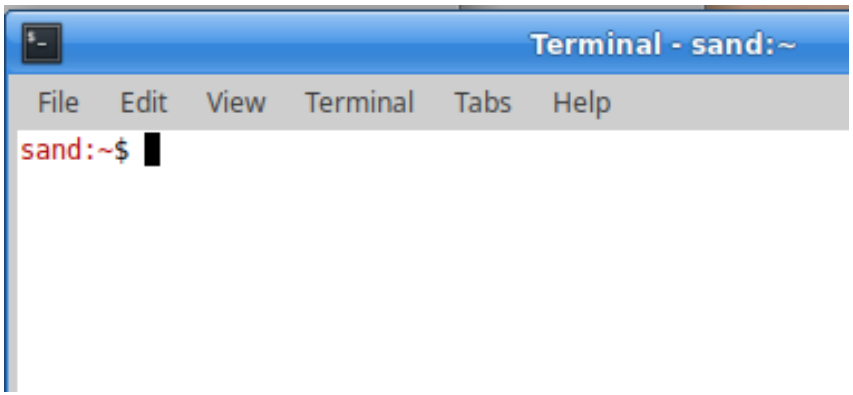
  - using the Applications menu in the upper left corner, do Applications → Terminal Emulator



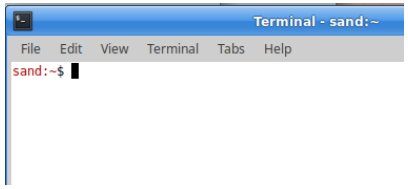
  - use the XFCE keyboard shortcut Ctrl-Alt-T to run the Terminal Emulator

Open a Terminal Window

- any of these should result in a terminal window containing the BASH command prompt denoted by a dollar sign $



The Command Prompt



- throughout the semester, this dollar sign is where you enter shell commands
- shell commands are the primary way for the user to communicate with the Linux operating system
- you are using a GUI window manager, so some things you can do with the mouse and menus
- but BASH is a Turing-complete interpreted programming language
- it is more powerful, efficient, and flexible than any menus

Commands

- a command line has the general form

```
$ command [options [option arguments]] [command arguments]
```

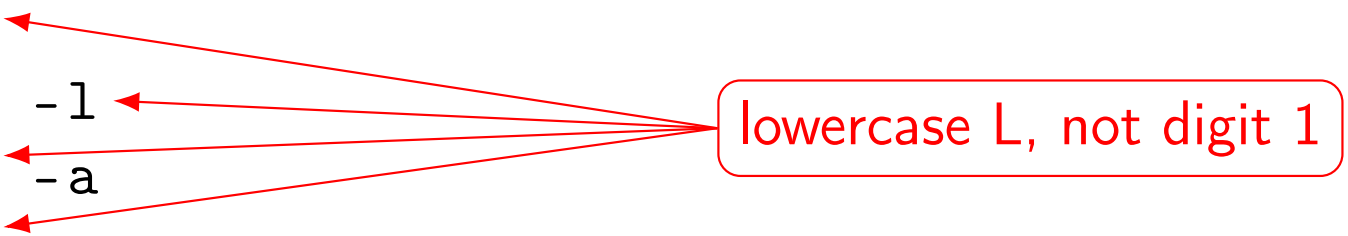
- where $ is the command prompt; you don't type it
- command is the name of the command
- stuff enclosed in square brackets may or may not appear, depending on the command

- an option may be preceded by zero, one, or two hyphens
- an option may or may not have arguments
- the command itself may or may not have arguments
- the command line is submitted to the BASH interpreter by pressing the Enter key

Examples

- issue each of these commands at the prompt and see what happens

```
1  $ ls
2  $ ls -a
3  $ ls -a --sort=size
4  $ ls -l
5  $ ls -a -l
6  $ ls -l -a
7  $ ls -la
8  $ ls .
9  $ ls ..
10 $ ls foobar
```



Example Explanations

`$ ls`	the list command, no options, no arguments; the default is to show the working directory
`$ ls -a`	the all option, indicated by the hyphen
`$ ls -a --sort=size`	the sort option, indicated by two hyphens, and with one option argument
`$ ls -l`	the long option
`$ ls -a -l`	two options
`$ ls -l -a`	same two options; order doesn't matter
`$ ls -la`	same two options, combined
`$ ls .`	no options, one argument; the period indicates the working directory
`$ ls ..`	two periods indicates the parent of the working directory
`$ ls foobar`	one argument, which presumably doesn't exist, and so we get an error message

Command Options

- another name for option is switch
- usually (but not always):
  - one hyphen precedes a one-letter option, also called a short option
    `$ ls -l`
  - multiple short options may be strung together if they have no arguments
    `$ ls -altrh` has five options
  - two hyphens are used for long options; these cannot be strung together
    `$ ls --sort=size --ignore-backups`
- some options have a long and a short form
  `$ s --ignore-backups` and `$ ls -B` are identical
- almost always, a short option with an argument may optionally be preceded by a space: `$ ls -w30` or `$ ls -w 30`
- usually, a long option with an argument must use equal sign with no spaces: `$ ls --width=30`

Files

A Central Tenet of Unix

Everything in Unix is a file.

- a file is simply a collection of bytes
- there are two main kinds of files in Linux

  plain file a plain file is a collection of bytes that is interpreted as data

  directory a special type of file that can contain other files, including other directory files

The Linux Filesystem

- the Linux filesystem is organized as a single unified tree-structured hierarchical structure
- the root of the filesystem is denoted as the slash (/)
- slash is a directory (which is a file that can contain other files)
- slash is the zero-level master directory of the Linux system
- to see the first-level contents of slash, use the command:
  `$ ls /`

The Contents of Root (/)

```
$ ls /
```

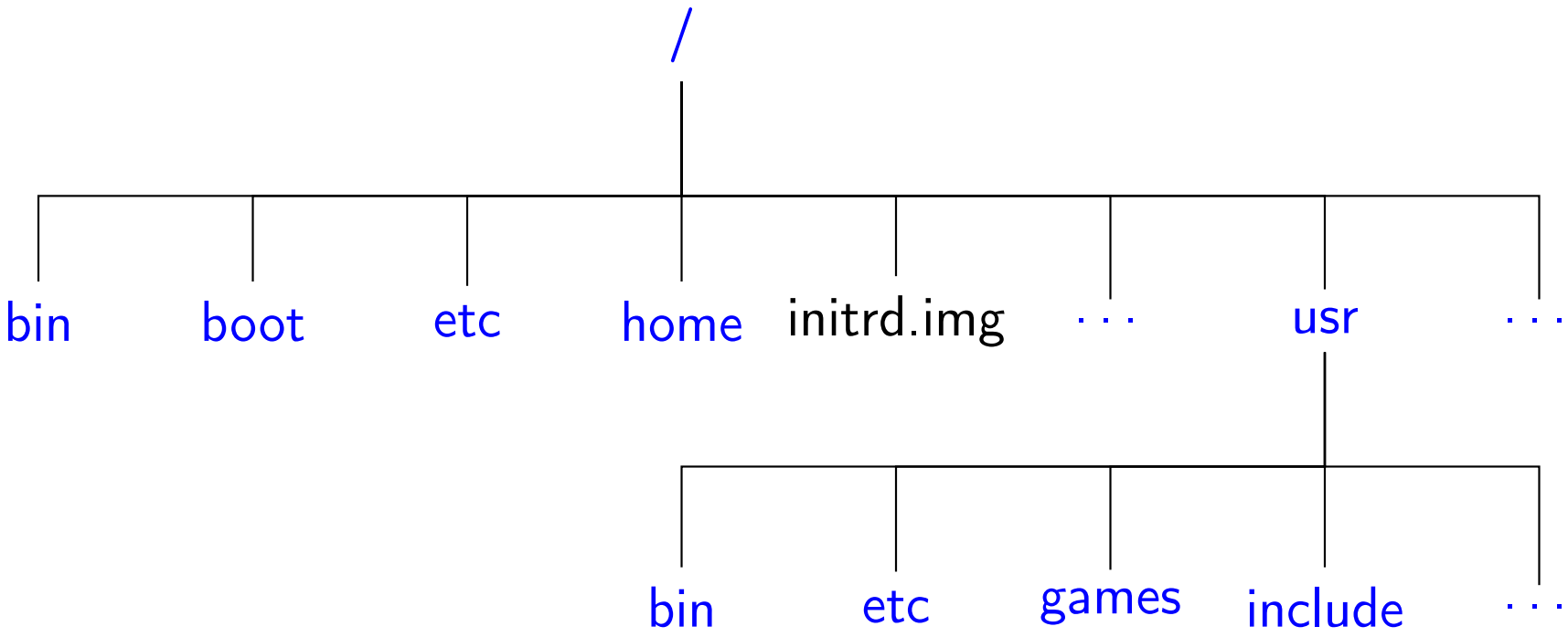
boot
dev
etc
home
initrd.img
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
snapshot
srv
sys
tmp
usr
var
vmlinuz

The Contents of /usr

```
$ ls /usr
```

bin
etc
games
include
lib
lib32
libexec
libx32
local
sbin
share
src

Viewed as a Tree



- directories in blue
- plain files in black
- names must be unique within a directory but not across the hierarchy

Current Working Directory

- for the shell, every command is in the context of a location in the filesystem: the current working directory
- print the working directory: `$ pwd`
- change the current working directory: `$ cd pathname`
- the argument can be any directory path name, relative or absolute e.g.,
  `$ cd /etc`
- the cd command with no argument changes the current working directory to your home directory

Absolute and Relative Pathnames

- for every file, there is a unique path from the root to that file
- the path can be specified as absolute or relative
- absolute
  - an absolute pathname always starts at the root (slash)
  - for example, `/usr/lib/spim/exceptions.s`
- relative
  - a relative path starts "here", the current working directory e.g., `lib/spim/exceptions.s`
  - or go up two levels, over and down `../../lib/spim/exceptions.s`
  - no leading slash on relative paths

Managing Directories

- directory organization is crucial to effectively use a computer
- on my laptop, just within my home directory, there are 449,362 files in 19,093 directories

- make a new directory: `$ mkdir name`
- WARNING! never use a space or special characters in directory names; you will regret it
- use only letters, digits, underscores, and sometimes periods

- if a period is the first character of a file (or directory) name, that is a hidden file
- it does not show in a ls listing, unless you include the -a (all) option

Rename a Directory

- the command to rename a file or directory is mv, which is a mnemonic for move, e.g.,
  `$ mv foo bar`
  which renames foo to bar
- this may seem funny
- but mv can also "rename" a file into a different directory, e.g.,
  `$ mv foo ../foobar/bar`
- and so move is the correct concept

- mv can also move multiple files into a single directory, e.g.,
  `$ mv foo1 foo2 foo3 bar`
  foo1 has now become bar/foo1
  bar must be a directory and must already exist
- can move files or directories

Deleting Files and Directories

- one or more files can be removed with rm:
  `$ rm foo bar`
- one or more empty directories can be removed with rmdir:
  `$ rmdir foo bar`
- if the directory is not empty, this fails

- to remove a non-empty directory, cd into it, remove its contents, then remove it with rmdir
- or, use rm with the -r (recursive) option:
  `$ rm -r foobar`

Warning!

You can delete a lot of stuff with the -r switch! Be careful.

Warning!

There is no undo for rm: it is instant and permanent!

- to be a little safer, you can use the -i (interactive) option