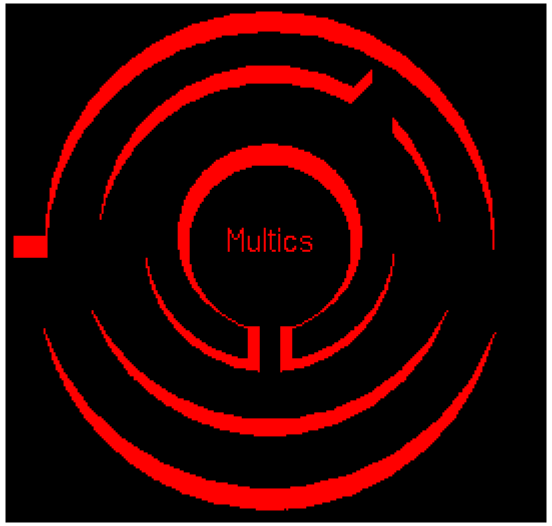
Linux Tutorial

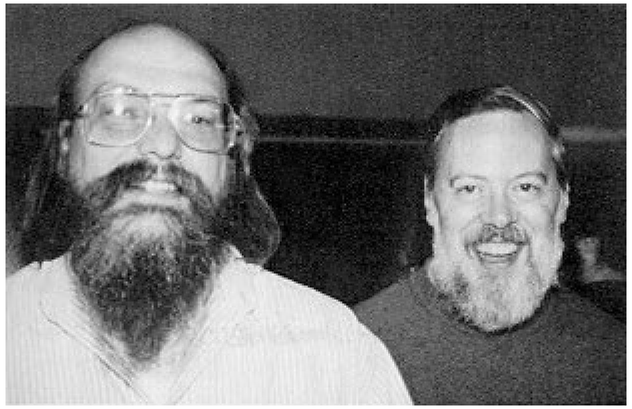
BY: EDUARDO FEO

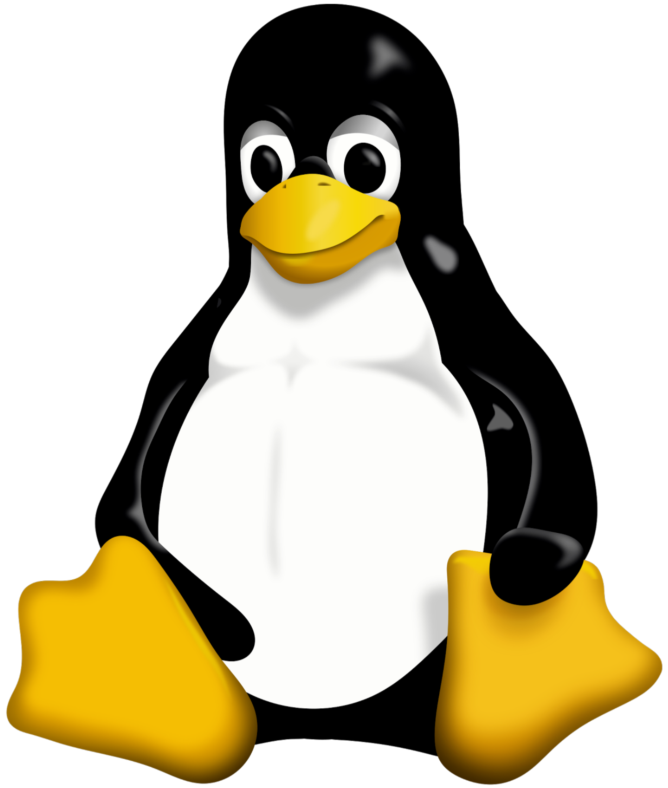
BASED ON ALEX STANESCU'S SLIDES

What is (GNU)Linux?

- Linux is the kernel
- Many Linux-based OS (Ubuntu, Debian, Red Hat)
- Like Windows, MacOS, Android, etc…

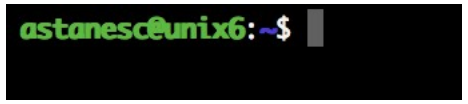






The Terminal

- In Linux, we generally use a text-based program called the *terminal* to run programs, edit files, and generally do everything we need to do
- The terminal allows us to interact with the OS on a deeper level than graphical interfaces.
- The terminal looks something like this:

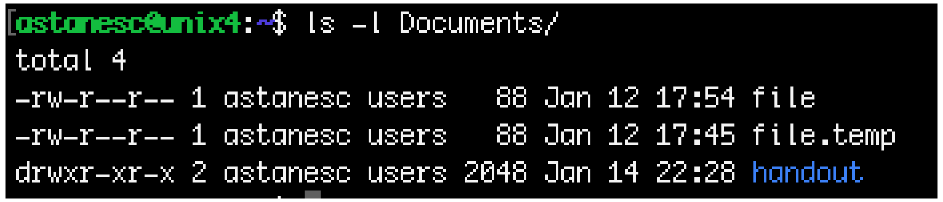


```
astanesc@unix6:~$
```

- Notice the **$**. Whenever we show commands, we usually include either a **$** or a **%** (called the prompt). ***DO NOT WRITE THIS IN THE TERMINAL***.

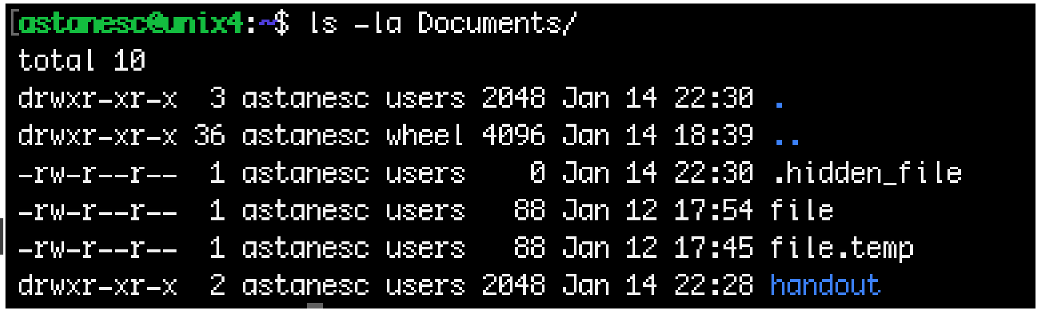
The Anatomy of A Command

- Let’s take a command and break it down to better showcase it



```
astanesc@unix4:~$ ls -l Documents/
total 4
-rw-r--r-- 1 astanesc users   88 Jan 12 17:54 file
-rw-r--r-- 1 astanesc users   88 Jan 12 17:45 file.temp
drwxr-xr-x 2 astanesc users 2048 Jan 14 22:28 handout
```

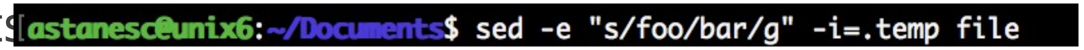
- The first part (**ls**) is the actual name of the command
- Next, we have flags. Flags are denoted by the dash (**-**), and provide extra information for the command.
  - **-l** indicates that we should give the user more information (like the date last modified)
  - We can also give multiple flags by just adding them to the dash like in the example to the right
- The last part is the main argument to the command itself (in this case Documents/)



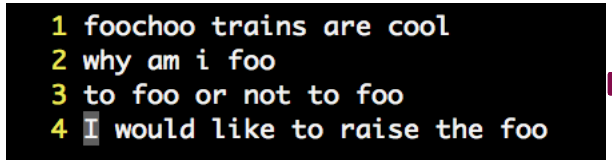
```
astanesc@unix4:~$ ls -la Documents/
total 10
drwxr-xr-x  3 astanesc users 2048 Jan 14 22:30 .
drwxr-xr-x 36 astanesc wheel 4096 Jan 14 18:39 ..
-rw-r--r--  1 astanesc users    0 Jan 14 22:30 .hidden_file
-rw-r--r--  1 astanesc users   88 Jan 12 17:54 file
-rw-r--r--  1 astanesc users   88 Jan 12 17:45 file.temp
drwxr-xr-x  2 astanesc users 2048 Jan 14 22:28 handout
```

The Anatomy of a Command (con't)

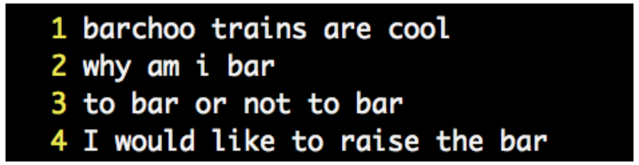
- Commands can take multiple arguments (see grep later on this page)
- Some flags can themselves take arguments



```
[astanesc@unix6:~/Documents$ sed -e "s/foo/bar/g" -i=.temp file
```

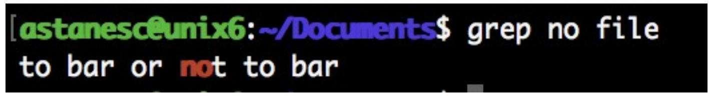


```
foochoo trains are cool
why am i foo
to foo or not to foo
I would like to raise the foo
```



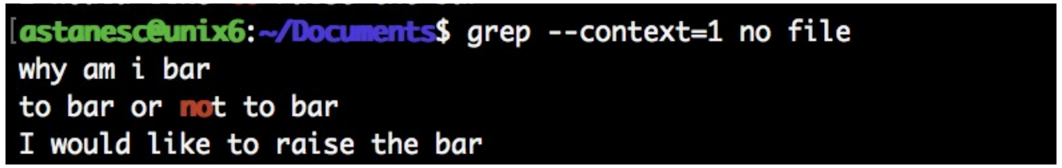
```
barchoo trains are cool
why am i bar
to bar or not to bar
I would like to raise the bar
```

- Some flags are not represented by just one letter



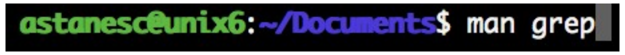
```
[astanesc@unix6:~/Documents$ grep no file
to bar or not to bar
```

vs.



```
[astanesc@unix6:~/Documents$ grep --context=1 no file
why am i bar
to bar or not to bar
I would like to raise the bar
```

More info on commands

- The **man** page for any given command contains more info about the command
- Accessed using the **man** command in terminal 

```
astanesc@unix6:~/Documents$ man grep
```

- The man page contains a lot of info
  - Description (What the command does)
  - Synopsis (How to use the command)
  - Options (What optional flags you can pass in)
  - You can even search for a **WORD** by typing **/WORD** and hitting enter
- Take a look at the man page for grep!
- Some commands also have a **-h** or **--help** flag that you can use

Navigating Your File System

- In Windows/Mac, we navigate the file system by clicking on folders and opening them.
- In Linux, we can do that as well, but it faster and typically more useful to use the terminal. Here “folders” are also referred to as “directories”
- In the terminal, there are three useful commands for navigating your filesystem
  - **cd** (**C**hange **D**irectory)
  - **ls** (**L**i**S**t files)
  - **pwd** (**P**rint **W**orking **D**irectory)

Special Directories

- There are five special directories
  - **/** = The root directory (i.e. the most upper-level directory)
  - **~** = The home directory (On afs this is something like /afs/andrew.cmu.edu/usr23/acarnegie)
  - **.** = The current directory
  - **..** = The previous directory (i.e. one level up)
  - - = Last working directory
- These directories can always be accessed at any point with commands like **cd** and **ls**
- These special directories can also be part of paths (e.g. **cd ~/private/15122/**)

Affecting Files

- There are many useful commands for creating, editing, removing, etc files. A shortlist is below:
  - **rm** – Removes the given file (**rm file.txt**) USE WITH CARE
  - **cp** – Copies the given file to a different location (**cp oldloc newloc**)
  - **mv** – Moves the given file to a different location (**mv oldloc newloc**)
  - **mkdir** – Makes a directory at the given location (**mkdir newdir**)
  - **grep** – search the given file for a string (**grep ”string” file**)
- More info can be found by googling a command or in the **man** pages for each command

Transferring Files to/from AFS

- For this, we generally use a command called **scp** (**S**ecure **C**o**p**y). This takes in two arguments, the original file to transfer, and the location to transfer to (in that order)
- For instance, if I wanted to transfer a file from AFS to our computer, I'd run **scp astanesc@unix.andrew.cmu.edu:private/15122/testfile.txt ./**
- I could also rename the file while transferring by giving the destination as a file (i.e. give it a name) rather than as a directory. For example, I could run **scp astanesc@unix.andrew.cmu.edu:private/15122/testfile.txt newfile.txt**
- On the other hand if I wanted to transfer a file from my computer to AFS, I'd run **scp testfile.txt astanesc@unix.andrew.cmu.edu:private/15122/**

Compressed Files

- On Windows, you may be familiar with the **.zip** extension for compressed archives (where an archive is just a collection of files with some associated metadata)
- On Linux, two different extensions are used two represent the compression and the archive.
  - A **.tar** archive is a group of files with some associated metadata
  - A **.gz** file is a g-zipped file – a compressed file. This is similar to **.zip**, but it is not an archive, but simply a file
  - Combined, these two form a **.tar.gz** or simply, a **.tgz** archive, which is a compressed archive (like **.zip** in Windows)

Compressing/Uncompressing Files

- In this (and future) CS classes, handouts and handins are given as compressed archives (a **tgz** file).
- To compress/uncompress these, we use the command **tar**. Tar supports both compressing and uncompressing.
- To compress, we run **tar –czvf archive.tgz file1 file2 file3 …** where **file1, file2, file3 …** are the files to compress, and **archive.tgz** is the file that we are creating. The flags are:
  - **c**: Compress
  - **z**: G-zip the archive . If you just want to just create a tar file, leave this argument off
  - **v**: Verbose (show every file that we are compressing)
  - **f archive.tgz**: Write to the file archive.tgz

Compress/Uncompress Files

- To uncompress, we simply change the **–c** to a **–x** (for expand). Thus, in the previous example, we would change the command to **tar -xzvf archive.tgz**
- **Note:** For whatever reason, some browsers (for example chrome) automatically unzip **.tgz** files without notifying the user. If you download a .tgz file, make sure to leave off the **–z** flag.
- See the **man** page for **tar** for more info
- You can also use **file** to determine the file type

EMACS

- Over 10,000 built-in commands
- LISP
- Modifiers:
  - C-   Control
  - M-   Meta
  - S-   Shift
- Some examples
  - C-d: calls delete-char
  - C-% search and replace
- Some of the emacs commands also work in the shell!
  - C-r and C-s to navigate the bash history

Speeding up your work

- Use <TAB>, <TAB>, ..., <TAB>
  - Can be used to autocomplete files, the command itself, directories and most anything

Speeding up your work

- Bash history
  - Up/down arrow navigates through your history
  - Search past commands: CTRL+r search_term
  - Reuse the previous command in present command with !!
  - Reuse the last item from the previous command with ALT+.

Speeding up your work

- Use **less** to read a file
- Use **grep <search_term>** to search within files
- Use **find** to search files within directories
- Redirect **input** and **output** with < and >

Speeding up your work

- Multiple commands
  - command_1; command_2; …
  - command_1 && command_2 && …

Speeding up your work

- Pipes: connect input and output in a sequence of commands
  - command_1 | command_2

Final Remarks

- Stuck?
  - Read Error Messages
  - Check the man page
  - Google it!
  - Ask on Diderot
  - Ask a friend
  - Come to office Hours!