# GEOS F436/636 Beyond the Mouse

Christine (Chris) Waigl
University of Alaska Fairbanks – Fall 2018
Week 10: The Unix command line 2 - scripting

## Topics for week 10

- Review of points from Week 9 - Unix command line 1
- More useful tools to get information (file, identify, ifconfig)
- More useful tools to communicate with remote systems (ping, ssh, scp, wget or curl)
- Text processing tools you may want to learn about (grep, sort, sed, awk, vi/vim
- Intro to shell scripting!

# Unix commands 6: more on getting information

```
$ file file1          ←  determine file type/information (really useful!)
$ identify file1      ←  file information from imagemagick suite of
                         image manipulation programs (comes with Ubuntu)
$ du -h               ←  get disk usage information in human-readable form
                         also a good option: --max-depth 1 (go 1 directory down)
$ env                 ←  print out all variables available in your environment
$ which command1      ←  find the path from which command1 is executed
$ ifconfig            ←  get networking information
```

If software isn't installed yet, you can install it from the command line:

```
$ sudo apt-get install net-tools   ←  sudo = "do this as the superuser"
```

## Unix interlude 4: Environment variables

These are variables that are available either just for a script (see later), or in general for all the software you run. Variables are referred to with $ before the name. You can find their value using echo $VARIABLE. Most are all-uppercase.

| | |
|---|---|
| `$HOME` | ← the path of your home directory; eg `/home/chris` |
| `$USER` | ← your currently logged-in username.; eg `chris` |
| `$PATH` | ← a list of directories where Unix looks for commands to execute<br>!! Important: a command is just an executable file on your PATH !! |
| `$SHELL` | ← the name (and path) of your default shell; eg. `/bin/bash` |

You can set your own variables in the file `.bashrc`, which is executed each time a bash shell is started.

# Unix commands 7: Communicate w/ remote systems

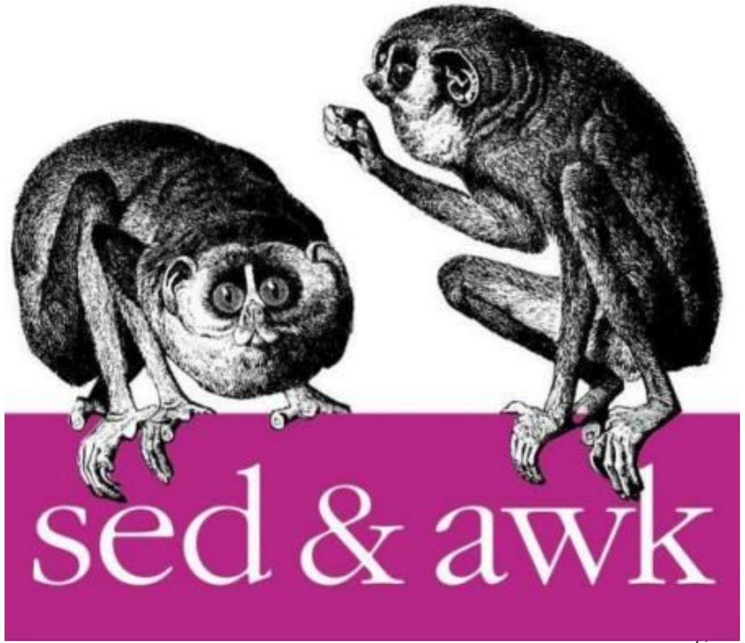
```
$ ping hostname (or: IP addr) ← check round-trip time across the internet
$ dig hostname             ← find out which IP addr is associated with hostname
$ dig -x IP addr           ← find out which host/owner is associated with IP addr
$ ssh user@host.domain.suffix ← securely connect to remote system's shell
$ scp filepath user@host.domain.suffix:/path/to/dest    or
$ scp user@host.domain.suffix:/path/to/file dest ← transfer files
$ wget http://url (or ftp://location) ← retrieve files from public server
```

(A similar command to `wget` is `curl`.)

# Unix commands 8: A few complex tools for text manipulation you might want to learn how to use

Unix has a lot of very powerful tools that amount to small programming languages, particularly useful in combination with pipes and redirection:

- `sort:` sort output; eg. `ls -a | sort -r` (sort directory list in reverse order)
- `sed:` a stream editor. Edit files line-by-line, to make substitutions; eg. `cat poem.txt | sed 's/wind/storm/g'` (s=substitute, g=globally)
- awk: a text processing scripting language. `awk options 'program' file` , for example: `awk '{print $1}' poem.txt` For more, see https://likegeeks.com/awk-command/



## Download some code from GitHub

Either use Firefox to go to
`https://github.alaska.edu/Fall2018-BtM/BtM2018_Linux`
... and select Clone or Download > Download ZIP. This will download the file `BtM2018_Linux-master.zip` to your ~/`Downloads` folder. Use `mv` to your home directory, then unzip it using `unzip BtM2018_Linux-master.zip .`
Alternatively, in your home directory do:

```
curl -u UAUsername -H "Accept: application/vnd.github.raw" --output master.zip -L
"https://github.alaska.edu/Fall2018-BtM/BtM2018_Linux/archive/master.zip"
```

1. **Explore your downloaded files. How many files are there in each directory? What type of files are they?**

## Shell scripting 1: Script files, running scripts

- A shell script is nothing but a text file that contains a list of shell commands that are executed one by one, usually with the extension `.sh`
- The first line ("hashbang" or "shebang" line) indicates which shell should be used to run the command: `#!/bin/sh` ← regular Bourne shell ; `#!/bin/bash` ← Bourne Again Shell `(#! = hash bang)`
- To execute a shell script, either make the file executable and run it, or use `sh script.sh`

## Shell scripting references

There are many good tutorials and books (usually by the publisher O'Reilly) available. Online resources:

- https://www.shellscript.sh/ ← for general shell scripting (Bourne shell)
- https://www.tldp.org/LDP/Bash-Beginners-Guide/html/index.html (Bash scripting)

It is fine to start out with the Bourne shell. If one day you want to write longer and more complex scripts, bash has a few features that makes it a little easier.

## Example for loop

```
#!/bin/sh
for i in 1 2 3 4 5
do
  echo "Looping ... number $i"
done
```

## Example IF statement

```
if  [ something ]; then
 echo "Something"
elif [ something_else ]; then
   echo "Something else"
else
   echo "None of the above"
fi
```