Pioneer AWS Linux Training

Housekeeping

- THERE ARE NO BAD QUESTIONS
- Intros
  - Name
  - Experience
- Remember there is almost always more than one way to do something

Prerequisites (if you want to follow along)

- An installation of Mint/Ubuntu/Debian (macOS will work as well)
- Access to a bash shell

Some basics

- What is Linux?
  - Linux is just the kernel, the interface to the hardware
  - A distribution (e.g. Ubuntu / Mint) is all the stuff that we use to do work packaged up conveniently (more or less)
- The file system starts at / (and it is all downhill from there)
- /var is (generally) where the system writes logs
- /etc is (generally) where config files are kept
- /usr is (generally) where libraries and applications are stored
- /proc is a funny imaginary directory with a bunch of cool stuff like a directory for every process

Navigating basics

- ls [directory or file]
  - List files in a directory
- ls -lah [directory or file]
  - Common usage details; all files; human readable
- ls -ltr [directory or file]
  - Common usage details; sort by time; reverse the sort order
- cd [directory]
  - Change directory
- pwd
  - Prints the current working directory
- The shortcut . references the current directory
- The shortcut .. references the parent directory
- The shortcut ~ references your home directory

Navigating knowledge check

- What command(s) would you use to
  - move to the directory one level up and list the contents?
  - list the contents of your home directory?

Navigating knowledge check (my way)

- What command(s) would you use to
  - move to the directory one level up and list the contents?

    ```
    cd ..; ls
    ```
  - list the contents of your home directory?

    ```
    ls -ltr ~
    ```

File permissions

- Every file has three groups of three permissions
  - What everybody can do to the file (or directory)
  - What the file (or directory) group can do
  - What the file (or directory) owner can do
- The permissions in the groups have letters and a “score”
  - read=r (4 points)
  - write=w (2 points)
  - execute=x (1 point)
- Other attributes of a file or directory
  - Type: directory=d; regular=-; symbolic link=l
  - Owner: who owns the file or directory
  - Group: what group of users are associated with the file

File permissions (continued)

- The permissions can be changed with chmod
    - For each of the three groups just add up the score of what you want it to look like
    - Example: you want a file to be readable and executable by only you, 4 + 1 = 5
        - chmod 500 something.sh
    - Example: you want a file read/write for yourself and readable for everybody else; 4 + 2 = 6, 4 = 4, 4 = 4
        - chmod 644 another.txt
    - Any usable directory needs the executable 1
- Other attributes of a file or directory
    - Type: directory=d; regular=-; symbolic link=l
    - Owner: who owns the file or directory
    - Group: what group of users are associated with the file

Permissions knowledge check

- How would you
    - make a file read only by the current user?
    - make a file readable by all users and writeable by only the owner?
    - make a file readable, writable, and executable by everyone?

Permissions knowledge check (my way)

- How would you
  - make a file read only by the current user?

    ```
    chmod 400 somefile.txt
    ```

  - make a file readable by all users and writeable by only the owner?

    ```
    chmod 644 another.txt
    ```

  - make a file readable, writable, and executable by everyone?

    ```
    chmod 777 harmless.sh
    ```

File system manipulation

- cp *<source> <destination>*
  - Copies a file from one location to another. The source file remains unaltered. More than one source can be listed. Works with wild cards (*)
  - Use –r for moving directories
- mv *<source> <destination>*
  - Similar to copy but moves the source file. Think of it as a Cut operation in windows. More than one source can be listed. Works with wild cards (*)
- rm <some-file>
  - Removes the file listed as the target. More than one target can be listed. Works with wild cards (*)
  - -r for directories. –f for read only files (BE CAREFUL!)
- mkdir *<new-directory>*
  - Makes a new directory at the target path
  - mkdir -p <directory-tree> makes a stack of directories
    - e.g. `mkdir –p a/b/c`

Viewing file contents

- cat <some-file>
  - Outputs the contents of a file to the current shell
- less *<some-file>*
  - Views file contents in current shell and supports paging. More is old but still used heavily, less is an updated version of more
- tail *<some-file>*
  - Used to view the last 10 lines of a file.
- tail -n 23 <some-file>
  - See the last 23 lines of the file
- head <some-file>
  - See the first 10 lines of a file
- head -n 23 <some-file>
  - See the first 23 lines of a file

File knowledge check

- How would you:
  - Copy a file from directory `a` to directory `b` and then delete the file in directory `a`?
  - View the last 50 lines of a file?
  - Move all files from directory `a` that have a .txt extension to directory `b`?
  - Print all lines of a file to the console?

File knowledge check (my way)

- How would you:
  - Copy a file from directory `a` to directory `b` and then delete the file in directory `a`?

    ```
    cp a/somefile.txt b/.
    ```
  - View the last 50 lines of a file?

    ```
    tail -50 b/somefile.txt
    ```
  - Move all files from directory `a` that have a .txt extension to directory `b`?

    ```
    mv a/*.txt b/.
    ```
  - Print all lines of a file to the console?

    ```
    cat b/somefile.txt
    ```

Files and input

- echo “some text”
  - Prints the provided text back to the screen
  - Can also print variables when executed in a script or from environment
- touch <*some-file*>
  - Creates an empty file at the target path or updates the modification time of an existing file
- something.sh > output.txt 2> error.txt
  - The > operator writes the output of the command to the file listed overwriting the current file contents
  - The “2” indicates stderr for something.sh
- something.sh >> output.txt
  - The >> operator appends the output of the command to the file listed

Files and input knowledge check

- How would you:
  - Write the text “this is my sample text” to a new file?
  - Write the text “this is my appended sample text” so that it is appended to an existing file?
  - View the contents of the appended text file?

Files and input knowledge check (my way)

- How would you:
  - Write the text “this is my sample text” to a new file?

```
echo “this is my sample text” > newfile.txt
```

  - Write the text “this is my appended sample text” so that it is appended to an existing file?

```
echo “this my appended sample text” >> newfile.txt
```

  - View the contents of the appended text file?

```
cat newfile.txt
```

Other utilities

- *command* | *command*
  - The | (pipe) operator sends the output of the first command as input to the second command
- sort
  - Sorts lines of input file or input sent via pipe
- wc
  - Prints number of lines, number of words, and characters
- env
  - Prints all environment variables for the current shell
- ps
  - Displays a list of all running processes for current user. Add aux to see all processes
- wget a “simple” file downloader
  - `wget http://www.google.com`
- curl used to make HTTP request from the shell
  - `curl -v http://www.google.com > google.html`

Other utilities continued

- top
  - See running processes and server resource utilization
- tar
  - Used to extract or create tar files.
  - Extract .gz file: tar xzvf *filepath*
  - Extract .bz2 file: tar xjvf *filepath*
  - Create .gz file: tar cvzf compressedfilepath pathtofiles
- grep
  - Used to search for text in files or input text
- Some editor and there are many choices. Learning `vi` or `emacs` is way beyond the scope of this little talk
  - `nano` is pretty simple text editor
  - `vi` is VERY powerful but hard to learn
  - `emacs` is also VERY powerful and differently hard to learn (note: I’ve never used `emacs`)

Utilities knowledge check

- How would you:
    - Get the count of the number of processes running on the system?
    - Get the HTTP response from google.com?
    - List the current environment variables, sort them in alphabetical order, and then write to a local file?
    - Download a file from a remote server and then find a specific string in the file contents? How would you do this in one line?

Utilities knowledge check (my way)

- How would you:
  - Get the count of the number of processes running on the system?

    ```
    ps auxwww | wc -l
    ```
  - Get the HTTP response from google.com?

    ```
    curl -v http://www.google.com > /dev/null
    ```
  - List the current environment variables, sort them in alphabetical order, and then write to a local file?

    ```
    env | sort > sorted_env.txt
    ```
  - Download a file from a remote server and then find a specific string in the file contents? How would you do this in one line?

    ```
    wget http://www.ipchicken.com | grep IP
    ```

Package Managers

- Used to install software from remote repositories and make software installs easier
- Ubuntu, Debian, Mint
    - apt-get

- Language specific
    - Python - pip

Environment file

- In your home directory .bashrc contains all the environment variables for your shell that you can set at the start
    - Run on new shell/terminal
- /etc/profile has a bunch of stuff is set for all users

Profile and rc files – example settings

- alias *simple_cmd=‘complex_cmd’*
  - Used to create command shortcuts
- export *name=‘value’*
  - Used to set environment variables accessible by child processes
  - Environment variables can be referenced by prepending a $ character
  - Example - appending and prepending to existing variables
    - export PATH=/some/addtion:$PATH
    - export PATH=$PATH:/some/addition

.bashrc knowledge check

- How would you:
  - Create an alias to monitor the file /var/log/system.log as a new lines are written to the log file?
  - Append the path /usr/bin to the environment variable named my_env?

.bashrc knowledge check (my way)

- How would you:
  - Create an alias to monitor the file /var/log/system.log as a new lines are written to the log file?

```
echo “alias watchlog=’tail -f /var/log/system.log’”
>> ~/.bashrc
```

  - Append the path /usr/bin to the environment variable named my_env?

```
export my_env=${my_env}:/usr/bin
```

SSH, SSH Config, SCP

- ssh *remote_user@remote_server*
  - Usually a SSH key is required. The –i option facilitates this
  - Private SSH keys typically stored under ~/.ssh/
  - Keys must only be accessible to user (chmod 600)
  - -L used for local forwarding
  - -D used for dynamic forwarding
  - Make it simple with SSH config…
- ~/.ssh/config
  - Preconfigure connections into simple alias

SCP

- scp *source_server:source_path remote_server:remote_path*
  - Used to copy files between systems
  - if source_server or remote_server are not provided, localhost is used by default
  - Again, simple with SSH config

Job Control

- Processes you start can run (basically) in 3 states
  - foreground: in control of the shell
  - background: the process is running in the background and you can use the shell
    - e.g. `long_runner.sh \ >output.txt 2> error.txt &`
  - backgrounded and detached: with the nohup command you can put a really long process into the background and exit the shell not killing the process. Output will go to nohup.out
    - e.g. `nohup really_long_runner.sh &`
- Signals can be sent with ctl-c (kill) and ctl-z (stop until `fg`)

The End

- There is a lot to linux and this just barely scratched the surface
- Let me (or others) help as needed
- Google knows way more about everything than I do
- Questions?