Lecture 6 (1-26-26)

Shell Scripting (also reviewing networking)

Motivating Questions

1. How do you execute a shell script?
2. How do you use variables?
3. How do you substitute the output of a command?
4. How do you match based on patterns?
5. How do you conditionally execute commands?
6. How do you check if a file exists?
7. How do you do repeated execution?
8. How do you process command line arguments?
9. How do you read standard input?
10. How do you group commands into a function?
11. How are variables scoped?
12. How do you do (integer) arithmetic?
13. How do you handle signals?
14. How do you debug shell scripts?

Recap from last lecture

- remember student10@cse.nd.edu is a **hostname** or **domain**
- remember 127.0.0.1 is a special address called the **local host**
  - to qurery the ip address of a domain you can use the `host` command
    - EX: `host dredd.h4x04.space`
- if you do `ss -tlpn`
  - this command shows what ports are currently listening on local machine
    - t is tcp
    - l is for listening
    - p is for port
    - n is for number
- if you do `nmap -v -Pn dredd.h4x0r.space`
  - then you get all the ports listening on the specified machine
- you can have multiple **domains** mapped to the same **ip address**
  - you can also have the same **ip address** mapped to **multiple domains**

- if you do `nc www.google.com 80 + some other stuff` you can get the exact same html code chrome would return
- `curl –L dredd.h4x0r.space`
  - this command will follow any redirections to get to the specified domain

---

New stuff

- What is more important bandwidth or latency?

- when you are looking at GB/second -> this is **bandwidth**
  - how much you can carry over time
    - You can measure bandwidth by:
      - downloading a big file and timing how long it takes
      - `wget` does this, and it shows you the **bandwidth** at the end
- **latency** is the delay
  - the delay
  - you can use `ping –c 1 dredd.h4x0r.space`
    - this returns the **latency**
  - you can also use `traceroute dredd.h4x04.space`
    -this shows you all routers passed through on the way to the destination
    -something cool:
    - if you do `traceroute google.com`
    - it will show you what google machine you end up on
    - google names their machines based on nearest airport code

---

- `sftp student10.cse.nd.edu`
  - this presents you with a shell and you can transfer files over the network
  - to verify contents of the file are the same (assuming you have an identical file on your local computer)
    - you can do `sha1sum bashrc.txt`
    - this returns some **unique** hexcode depending on the **contents** of the same file
    - if the results match then we have the same **contents**

`tmux`

- life changing command: `tmux`
    - you can split the terminal into multiple screens
        - so you dont have to log in multiple times
- `tmux attatch` allows you to go back to where you were even if you disconnect from student machine

---

Shell Scripting

- vim hello.sh

```
#!/bin/sh
echo "Hello, World!"
```

- remember `.sh` files are not naturally executable
    - to make `hello.sh` executable you can do `chmod +x ./hello.sh`
        - **OR** you can do `sh ./hello.sh` or `bash ./helllo.sh`
            - **In our class we will use we will use:** `sh ./hello.sh`
        - **OR** you can just add the comment `#!/bin/sh` at the top of the script
            - the `sh` is the **schabang**
                - this is the program that specifies how the script is interpreted
                - we will use `sh` in our class
            - this specifies the **interpreter**

```
#!/bin/sh
echo "Hello, $NAME!"
```

- in this case you need to run `export NAME` for the script to be able to read Name
- or you can do this `NAME=${NAME:-Anonymous}`
    - this means that if `NAME` is defined, use that value, otherwise default to Anonymous
- you can change variable temporally by doing this:
    - `env NAME=sam ./hello.sh`

Example Script:

```
#!/bin/sh
NAME=${NAME:=-Anonymous}
DATE=`date`
```

```
case $NAME in
     carlos) GREETING=hola  ;;
     sam|sammy)    GREETING=sup.  ;; #you can also use wildcards like sam*)
     *)      GREETING=hello ;;
esac


echo "$GREETING, $Name!"
echo "Today is ${date}"
```