Process Management

EE 107S: Introduction to Linux

Lecture 6: 02/21/17

Background and foreground processes

- Before UIs, we needed to be able to run multiple processes concurrently
- So far everything we have run is a foreground process
- Can have up to one process in the foreground and as many in the background
- To run a command in the background, add & at the end

Seeing running processes: `jobs`

- Lists all background processes
- Either `running` or `suspended`
  - Processes can be suspended with `ctrl+z`
- Each process is given an ID

Bringing back processes: `fg`

- Picks the item in `jobs` with a + next to it to resume
- Job ID can be specified with %
  - `fg %2`

Resuming to the background: `bg`

- Works the same way as `fg`

Listing processes: ps

- Default behavior shows processes in current terminal
- Add -a to see other users' processes
- Add -x for more information
- Add -u to specify a user

Task manager: htop

- Output like a task manager
- Contains information on CPU/memory usage
- Can be displayed in tree form
- Can be used to kill processes

Terminating processes: `kill`

- Sends a “signal” to the process specified by PID
- Common signal is 9 for SIGKILL
  - `kill -9 12345`

Environment variables

- Variables set by the shell for all commands to access
- PATH keeps track of where executables are stored
- HOME has the same path is ~
- Can access environment variables with `echo`
  - `echo $PATH`
- Can set environment variables using `export` command (works in `.bashrc` too)

Crunchbang/shebang

- Must be first line in the file
- Tells the shell how it should execute the file
- `#!/bin/bash` at the top of `script.sh` means run the command:

```
/bin/bash script.sh
```

Variables

- Name followed by = followed by value (no spaces)
  - Value can be a number, a string, or the output of a command
    - `num=1`
    - `str='hello'`
    - `out=$(ls -l)`
- Refer to variables with `$varname`

Command line arguments

- Arguments are stored in special variables with `$number`
  - `$0` is the script name
  - Remaining arguments are 1-9 (no more than that as far as I know)
- `$#` contains number of arguments

If statements

```
if [[ condition ]]
then
   echo ”You made it”
else
   echo ”You didn’t make it :(”
fi
```

If statements

- Condition can be composed of various checks, some of which include
  - `INTEGER -eq INTEGER`
  - `STRING = STRING`
  - `-e FILENAME`
- Conditions can be combined with && or ||

While loops

```
counter=1
while [[ $counter -le 10 ]]
do
    echo $counter
    counter=$(($counter + 1))
done
```

For loops

```
for file in $(find . -type f)
do
     if [[ -e $file ]]
     then
          echo ”$file exists”
     fi
done
```

Reminder

- Near the end of the semester we'll send out an eCIS
- Please fill it out!

References

- Bash scripting tutorial