Unix Scripting

Week 8

Agenda

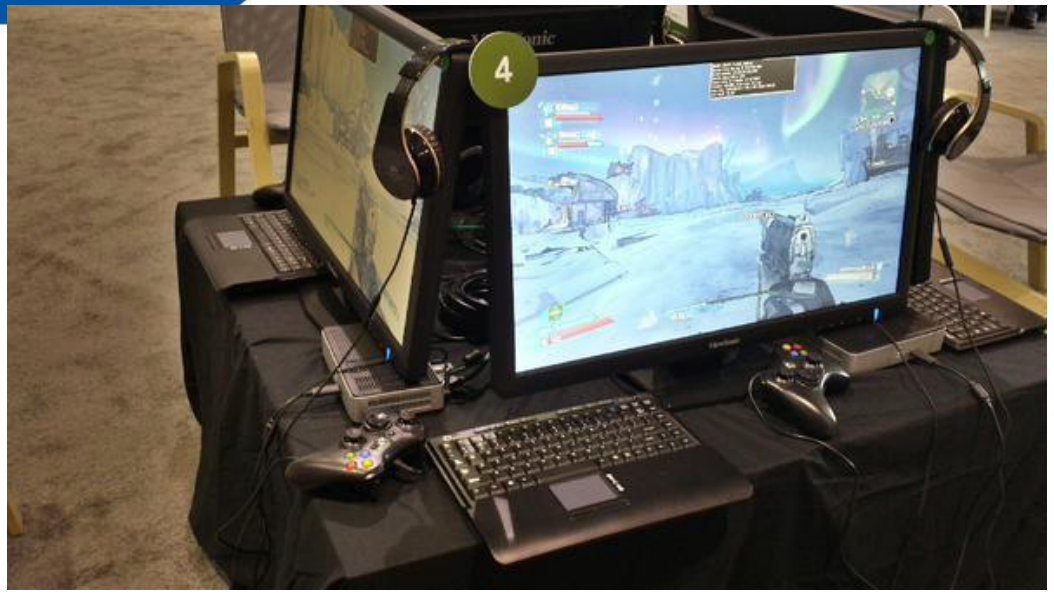
- Terminal Handling
  - Terminal settings and info
- Control structures in Unix
  - CASE…ESAC
  - SELECT….

Terminal

- a ***dumb terminal*** is basically a text display with a keyboard
  - real dumb terminals are largely a thing of the past, instead we emulate their function with programs such as telnet, ssh, putty
  - Unix is very good at handling many different kinds of terminals

Past: Mainframe computer and terminals

Current/Future: Cloud, SaaS, smart devices!



set terminal characteristics: **stty**

- In shell scripts you use the "stty" command
  - **stty**
    - will show an abbreviated list of terminal capabilities
  - **stty -a**
    - will show all terminal capabilities

STTY…

- **stty –echo**
  - disables echoing of typed characters
- an example of a simple "password" script:
  - `echo -n "Enter password: "`
  - `stty -echo`
  - `read password`
  - `stty echo`
  - `echo -e "\nYou entered '$password'"`

Example

```
oldsettings=$(stty -g)
stty -icanon min 1
key=
while [ "$key" != "q" ]
do
    echo -n "Hit any key: "
    key=$(dd bs=1 count=1 2> /dev/null)
    echo -e "\nYou hit the \"$key\" key"
done
stty $oldsettings
```

Collect Terminal info

- echo $TERM
  - TERM environment variable selects which terminfo entry to use for the shell
- **infocmp**
  - cub: cursor back
  - cuf: cursor forward
  - …
- **tput**
  - **tput cols:**gives number of columns in current display
  - **tput lines:** gives number of lines in current display

Terminal settings

- Traditionally, when you log into a Unix system, the system would start one program for you.
  - a Bourne shell, reads commands from ~/.profile when it is invoked as the login shell.
  - Bash reads commands from **~/.bash_profile** when it is invoked as the login shell, and if that file doesn't exist[1], it tries reading **~/.profile** instead
  - When you start bash as an interactive shell (i.e., not to run a script), it reads **~/.bashrc** (except when invoked as a login shell, then it only reads ~/.bash_profile or ~/.profile.

Terminal settings

- **~/.profile** is the place to put stuff that applies to your whole session, such as programs that you want to start when you log in (but not graphical programs, they go into a different file), and environment variable definitions.
- **~/.bashrc** is the place to put stuff that applies only to bash itself, such as alias and function definitions, shell options, and prompt settings. (You could also put key bindings there, but for bash they normally go into ~/.inputrc.)
- **~/.bash_profile** can be used instead of ~/.profile, but it is read by bash only, not by any other shell. (This is mostly a concern if you want your initialization files to work on multiple machines and your login shell isn't bash on all of them.) This is a logical place to include ~/.bashrc if the shell is interactive.

case

- Bash shell case statement is similar to switch statement in C.
- Syntax of bash case statement.
  - `case expression in`
  - `    pattern1 )`
  - `        statements ;;`
  - `    pattern2 )`
  - `        statements ;;`
  - `    ...`
  - `esac`

key points of bash case statements

- Case statement first expands the expression and tries to match it against each pattern.
- When a match is found all of the associated statements until the double semicolon (;;) are executed.
- After the first match, case terminates with the exit status of the last command that was executed.
- If there is no match, exit status of case is zero.

Activity: develop a script as follow and explain how it works

```
echo -n "Enter the name of an animal: "
read animal
echo "Here are some interesting facts about ${animal}s:"
case $animal in
   lion)  echo "Baby lions are cute"
          echo "Lions are generally scaredy-cats"
          ;;
   tiger) echo "Tigers have stripes"
          echo "Tigers are native to Detroit"
          ;;
   bear)  echo "Oh my!!!"
          echo "Bears are big and hungry and generally
not sociable"
          ;;
   *)     [[ $animal =~ ^[aeiou] ]] && n=n || n=
          echo "I don't know what a$n $animal is, but
I'm sure it's awesome!"
          ;;
esac
```

Warm up Activity1

What does the following commands do?

- `echo guide`tput bold`guide`tput sgr0`
- `echo `tput smul`guide`tput rmul``
- `tput -S << EEE`
  - `clear`
  - `cup 10 11`
- `EEE`

tput

- The **tput** utility uses the **terminfo** database to make the values of terminal-dependent capabilities and information available to the shell, to initialize or reset the terminal, or return the long name of the requested terminal type.

$()

- Command substitution allows the output of a command to replace the command itself.
- Command substitution occurs when a command is enclosed as follows:
  - `$(command)`
  - `Or `command``
- Bash performs the expansion by executing ***command*** in a subshell environment and replacing the command substitution with the standard output of the command, with any trailing newlines deleted.

select

- The select construct generates a menu from a list of items. It has almost the same syntax as the for loop:
  - **`select`** `ITEM in [LIST]`
  - **`do`**
    - `[COMMANDS]`
  - **`done`**
- The `[LIST]` can be a series of strings separated by spaces, a range of numbers, output of a command

Observation: run the following

```
# Define the menu list here
select brand in Samsung Sony  Apple
do
  echo "You have chosen $brand"
done
```

What is PS3?

- You can define a custom prompt for the select loop inside a shell script, using the PS3 environment variable
- Learn more about PS1, PS2, PS3, PS4
  - https://www.thegeekstuff.com/2008/09/bash-shell-take-control-of-ps1-ps2-ps3-ps4-and-prompt_command/