SCHOOL OF INFORMATION AND
COMMUNICATIONS TECHNOLOGY

Unix Scripting

Lecturer: Shahdad Shariatmadari

July 2020

Agenda

- Signals and Traps!

What is a Signal?

- ***Signals*** are software ***interrupts*** sent to a program to indicate that an important event has occurred.
- Some signals, such as the interrupt signal, indicate that a user has asked the program to do something that is not in the usual flow of control.
  - Like when you press **Ctrl+C**

list of common signals

Signal Name	Signal Number	Description
SIGHUP	1	Hang up detected on controlling terminal or death of controlling process
SIGINT	2	Issued if the user sends an interrupt signal (Ctrl + C)
SIGQUIT	3	Issued if the user sends a quit signal (Ctrl + D)
SIGFPE	8	Issued if an illegal mathematical operation is attempted
SIGKILL	9	If a process gets this signal it must quit immediately and will not perform any clean-up operations
SIGALRM	14	Alarm clock signal (used for timers)
SIGTERM	15	Software termination signal (sent by kill by default)

Signal?

- For learning more about signals, Run
  - `man 7 signal`

- For listing all signals, Run
  - `kill  -l`

Signal default action

- Every signal has a default action associated with it. The default action for a signal is the action that a script or program performs when it receives a signal.
- Some of the possible default actions are −
  - Terminate the process.
  - Ignore the signal.
  - Dump core. This creates a file called **core** containing the memory image of the process when it received the signal.
  - Stop the process.
  - Continue a stopped process.

Sending Signals

- There are several methods of delivering signals to a program or script.
  - One of the most common is for a user to type **CONTROL-C** or the **INTERRUPT key** while a script is executing.
    - When you press the ***Ctrl+C*** key, a **SIGINT** is sent to the script and as per defined default action script terminates.
  - The other common method for delivering signals is to use the **kill command**:
    - `kill -signal pid`
      - Here **signal** is either the number or name of the signal to deliver and **pid** is the process ID that the signal should be sent to.

Trapping Signals

- **Trap** allows you to **catch signals** and execute code when they occur.
  - `trap commands signals`
  - Here *command* can be any valid Unix command, or even a user-defined function, and signal can be a list of any number of signals you want to trap.

common uses for trap

- There are two common uses for trap in shell scripts :
  - Clean up temporary files
    - `trap "rm $WORKDIR/work1$$ $WORKDIR/dataout$$; exit" 1 2`
      - Now these files will be removed if the line gets hung up or if the *Ctrl+C* key gets pressed.
  - Ignore signals
    - `trap ''  2`
      - This specifies that the interrupt signal is to be ignored.

Activity: Step1

- Create a script in your matrix, name it *yourname_signal.sh* and run it. Explain how it works.

```
#!/bin/bash
echo "pid is $$"
while ((Count < 5))
do
  sleep 3
  (( Count++ ))
  echo $Count
done
exit 0
```

Activity: Step2

- Run the Script
- In the middle of execution, press Ctrl+C
- Edit the script and insert the following code in line2
  - `trap “You tried to Stop Me!”  2`
- Run the script again and press Ctrl+C
- What happen? Explain your observation!

Activity: Step3

- We are going to demonstrate how to kill a process by sending a signal
  - Run the script
  - Press Ctrl+Z
    - What happen?
  - Run : ps
    - What happen?
  - Run `kill -9 PID`
    - PID is the same number that is displayed
  - Run : ps
    - What happen?
- What happen? Explain your observation!