ULI101: INTRODUCTION TO UNIX / LINUX AND THE INTERNET

WEEK 8: LESSON 2

MANAGING PROCESSES
ALIASES AND COMMAND HISTORY

LESSON 2 TOPICS

Processes

- Process Definition / Foreground vs Background Processes
- Running Processes in the Background
- Managing Processes
- Demonstration

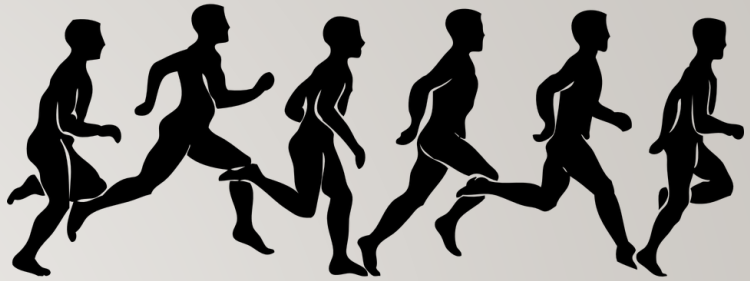
Aliases & Command History

- Purpose / Usage / Demonstration

Perform Week 8 Tutorial

- Investigations 2 and 3
- Review Questions (Questions 3 – 8)

MANAGING PROCESSES



Processes Definition

All programs (tasks) that are **running** on a Unix/Linux computer system are referred to as **processes**.

Characteristics of Processes:

- Each process has an **owner**
- Each process has a unique ID (**PID**)
- Processes keep their **PID** for their entire life.
- Usually a parent **sleeps** (i.e. **suspended**) when a **child is running** (the exception is when the child process is running in the background)
- UNIX / Linux processes are **hierarchical**. The process structure can have **children processes**, **great grandchild processes**, etc.

MANAGING PROCESSES

Viewing Process Information

You can issue Linux commands to provide information regarding running processes.
The **ps** (*process status*) command displays a **snapshot** of process information.

The **top** command provides **real-time** status of all running processes (press **ctrl-c** to exit top command)

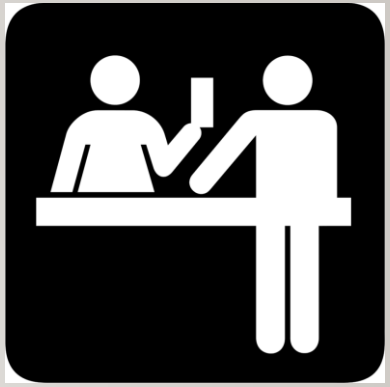
Linux Command	Purpose
`ps`	Basic listing of processes in current user's terminal, for example: **PID, process names**.
`ps -l`	Detailed listing in current user's terminal for example: **PID**, parent PID (**PPID**), **status**, etc.
`ps -ef`	Detailed listing ALL processes running on entire system.
`ps aux`	Detailed listing of processes for **ALL users** and background running services (i.e. **DAEMONS – background running services**).
`ps –U username`	Basic listing of processes running for a particular **user**.

MANAGING PROCESSES

Instructor Demonstration

Your instructor will now demonstrate how to **view** processes.

MANAGING PROCESSES



Foreground vs. Background Processes

Processes in UNIX can run in the **foreground** or **background**

Commands issued from the shell normally run in the **foreground**.

Programs / Commands can be run in the **background** by placing an **ampersand &** after the command.

For example: `command &`

MANAGING PROCESSES

Managing Foreground Processes

Users can **manage processes** to become more **productive** while working in the Unix / Linux Command-line environment.

Below are keyboard shortcuts to manage **foreground** processes.

Linux Command	Purpose
`ctrl-c`	**Terminates** a process running in the **foreground** interupt
`ctrl-z`	Sends a process running in the foreground into the **background**. Process is stopped (suspended) in background and requires **bg** command to run in background.

MANAGING PROCESSES

Managing Background Processes

Below are common Linux commands / **keyboard shortcuts** to manage **background** processes.

Linux Command	Purpose
`fg`	The **fg** (foreground) command moves a *background* job into the **foreground**. The fg command issued without arguments will place the most recent process in the background to the foreground. *Example:* `fg %job-number`
`bg`	The **bg** utility **resumes suspended jobs** from the current environment. The bg command issued without arguments will run the most recent process that was placed into the background. *Example:* `bg %job-number`
`jobs`	The **jobs** utility displays the status of jobs that were started in the current shell environment

Ctrl + z : Sends a process running in the foreground into the background. Process is stopped (suspended in background and requires bg command to run in background.

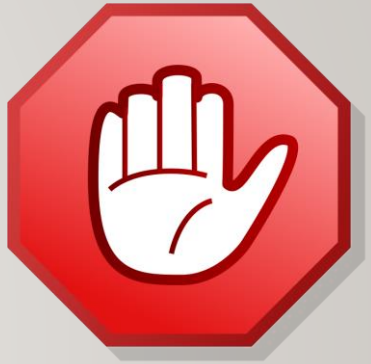
MANAGING PROCESSES

Instructor Demonstration

Your instructor will now demonstrate how
to **manage foreground** and **background** processes.

MANAGING PROCESSES

Terminating Processes



You can use the **kill** command to terminate processes.
You need to be the **owner** of the process to perform this operation.

The **kill** command sends the specified signal to the specified processes or process groups. If no signal is specified, the **SIGTERM** signal **(#15)** is sent.
The default action for this signal is to **terminate** the process.

If the TERM signal does NOT work, you can issue the kill command with the **option -9** (i.e. **SIGKILL, signal #9**).

Examples:

```
kill %jobnumber
kill -9 %jobnumber
kill PID
kill -9 PID
```

MANAGING PROCESSES

Instructor Demonstration

Your instructor will now demonstrate how to **terminate** processes.

ALIASES / COMMAND HISTORY

Using Aliases

Using the **alias** command assigns a **nickname** to an existing command or a series of commands. The **unalias** command is used to remove existent aliases.

Examples:

`alias` (alias command without an argument will display all the aliases currently set)

```
alias dir=ls
alias lal='ls -al'
alias clearfile='cat /dev/null >'
```

`unalias clearfile` (removes alias **clearfile** from memory)

ALIASES / COMMAND HISTORY

Command History:

The `~/.bash_history` file stores recently executed command lines.

There are several techniques using the `~/.bash_history` file to run previously-issued commands..

Examples:

`<up>` or `<down>`	move to **previous** or **next** command in Bash shell prompt
`fc -l`	display last **16** commands
`history | more`	display all stored commands
`!#`	**re-executes** command by command number (obtained from *history* command)
`!abc`	**re-executes** last command beginning with string "*abc*"

MANAGING PROCESSES

Instructor Demonstration

Your instructor will now demonstrate how to use **aliases** and **command history**.

HOMEWORK

Getting Practice

Perform **Week 8 Tutorial:**
(Due: Friday Week 9 @ midnight for a 2% grade):

- INVESTIGATION 2: MANAGING PROCESSES
- INVESTIGATION 3: ALIASES / COMMAND HISTORY
- LINUX PRACTICE QUESTIONS (Questions **3 – 8**)