LINUX
SYSTEM SUPPORT

(CST8207)

Course Intro

Algonquin College

Week1

By: Arsalan Parsaei

WHAT WE LEARN IN THE COURSE?

1. Understanding UNIX/UNIX like/GNU/Linux operating system
2. Use Linux command line (shell) to support user-level administration tasks, I/O redirection and List, set, unset, and use shell variables.
3. Use of patterns matching to selectively match pathnames
4. Set up a Linux machine on virtualized environment
5. File manipulation, File Access Control
6. Manage permissions in a GNU/Linux-based environment.
7. Script writing & debugging
8. Linux Automation
9. Linux Back up and restore , Comp/Decomp technology in Linux
10. Linux process managements

WHAT WE USE AS COURSE MATERIALS

- **Free Internet Resource**
- Optional: The bookstore may sell an optional CST8207 Textbook Package (ISBN: 0-132-37382-3)
- Optional: A Practical Guide to Fedora and Red Hat Enterprise Linux by Mark Sobell, Prentice Hall
- This course is part of the Bring Your Own Device (BYOD) program initiative at Algonquin College.
- Students are required to have a functioning laptop device (Windows machine) for completing course

HOW TO PASS THIS COURSE

- Making scheme is as follows:
  - Lab assignments (Weekly) (%30 total)
  - Mid-Term Exam (%25)
    - Mid-term 1 (%10)
    - Mid-term 2 (%15)
  - Final Exam written (%40)
  - Quizzes (%5)

PROFESSOR CONTACT

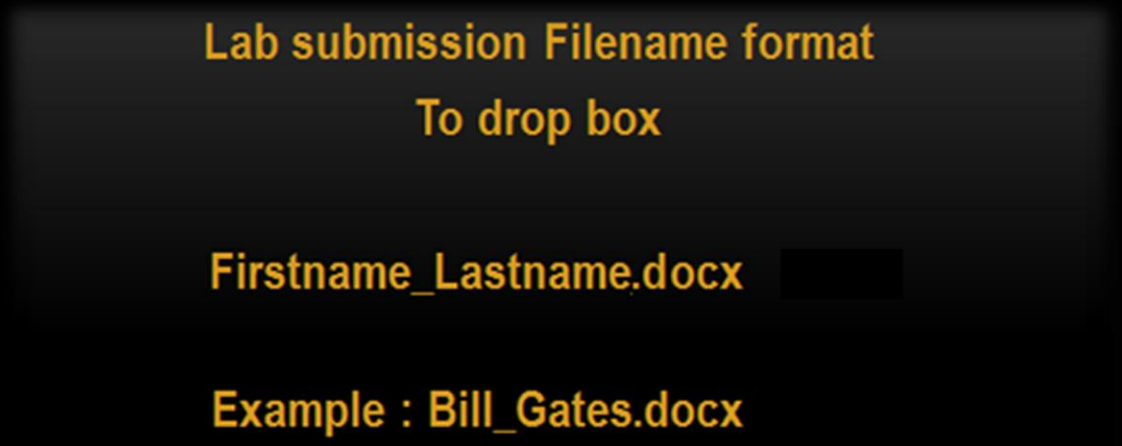
parsaea@algonquincollege.com

VIRTUAL OFFICE TIME:

Course page -> content -> Office Time

HOW TO SUBMIT YOUR LAB REPORT

- Your lab report must be submitted in DropBox in course page based on lab report instruction and guideline by professor
- Lab file name format must be as follow



WHY WE NEED OPERATING SYSTEM ?

WHAT IS VIRTUALIZATION FOR?

UNIX and UNIX “like”

Operating Systems

Unix Operating System (OS)

UNIX is :

- An operating system that has **been around since the 1960's**
- A **trademarked** name (www.unix.org)
- One that **conforms to a set of standards**, and is **registered** with the governing body of the trademark

We cannot call other similar operating systems a UNIX operating system

UNIX OS Examples:

Since the 1960's, many variations of the UNIX operating system have **come and gone**. The most notable (**currently active**) **UNIX** operating system platforms include:

- Hewlett Packard (HP-UX)

- IBM AIX

- Oracle Solaris

- BSD

What about Linux ? Is it same as UNIX?

How about Linux?

Can we call Linux is an Operating System ?

YES?

or

NO?

How about Linux?

Linux has **been around since the 1990**'s and it is a **trademarked** name (www.Linuxmark.org)

Note: Linux refers to the **kernel** ... and **not** the operating system as a whole

Therefore, any operating system that utilizes the Linux kernel is generally referred to as a **Linux "based" operating system**

Linux OS Examples:

In the Linux market there are many Linux distributions which some of them are gone and some still in the markets.
The more popular **Linux** operating system platforms include:

- **Red Hat** Enterprise Linux
- **CentOS** Linux
- Oracle Linux
- Ubuntu Linux
- Debian Linux
- Fedora

These are currently active

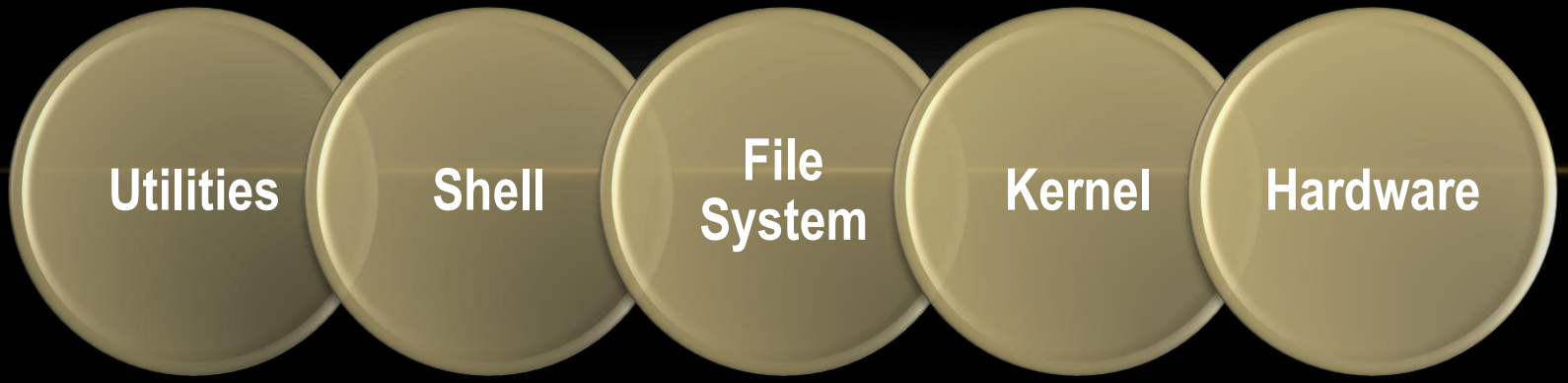
Linux kernel

- The **Linux kernel** by itself is basically useless to most people.
- As such, **Linux ”distributions”** are created which provide additional supporting software (along with the Linux kernel), to make a **more complete and functional operating system**
- There are **many Linux distributions** to choose from, with each having their own strengths and weaknesses
- **Google Android** is a Linux “based” operating system, and is found on the majority of smartphones worldwide

Major UNIX Operating System Components

Regardless of the **UNIX "flavor"** you are using (e.g Linux, BSD, Solaris, etc.) they all have **very similar functionality**

(4) of the **major components** in all UNIX and UNIX "like" operating systems versions are the:



4 or 5 components?

KERNEL

The **kernel** is the core (e.g. nucleus) of any UNIX or UNIX “like” operating system

The kernel **provides basic services** for the rest of the operating system by **supporting functionality** such as:

- **Multi-tasking**
- **File I/O**
- **Connectivity to networks and devices**
- **Resource allocation**

Often, the only major difference between (2) UNIX or UNIX “like” nodes, is **which kernel version** they run

SHELL

From user side the Shell important part of OS as it :

- Provides a command-line interface where the user can execute commands
- Interprets the user commands
- Passes the interpreted commands to the kernel

Go to your windows and find the **equivalent** to the UNIX or UNIX “like” shell in windows operating system.
(**cmd.exe**)
(**powershell.exe**)

Variant of UNIX Shell

There are several popular UNIX and UNIX “like” shells available

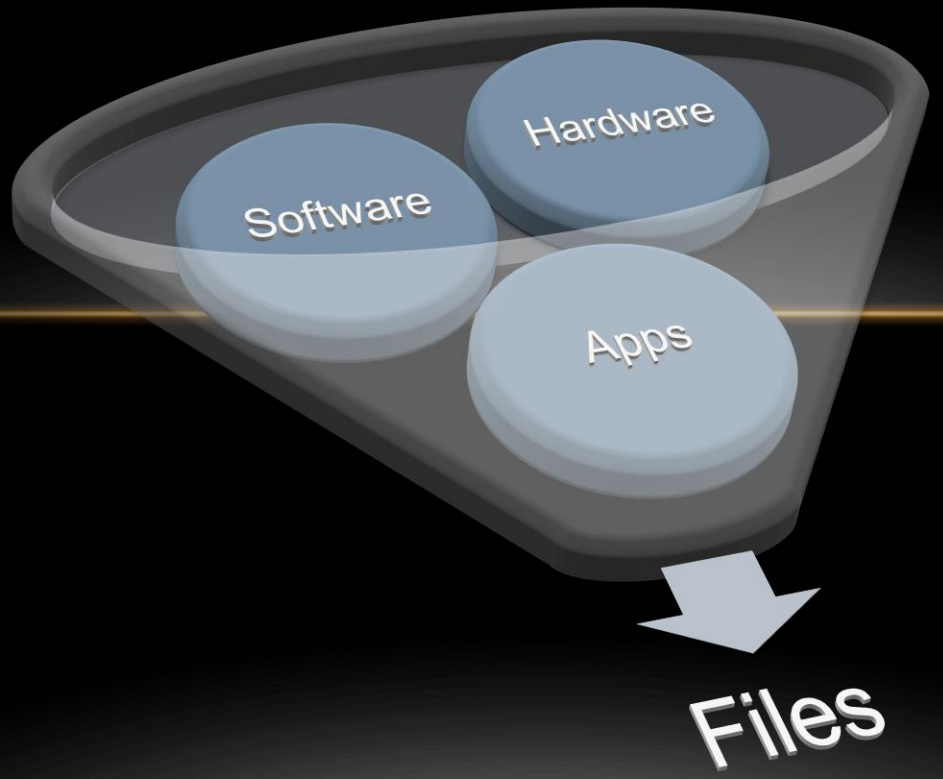
- **Bourne Shell (sh)**
- **Bourne-Again Shell (bash)**
- **C Shell (csh)**
- **Korn Shell (ksh)**
- etc.

(**bash**) is the lead in the shell market

UNIX/UNIX-Like Filesystems

“EVERYTHING is a file”

Applications, data, printers, hard disks, optical discs ... even a mouse ... are represented as files



LINUX BOX

- What is Linux Box
- Linux VM
- Hypervisor
- VMware
- VirtualBox
- Host OS
- Guest OS

THINGS TO DO FOR NEXT CLASS

- Finish Assignment 1 (Set up your Linux Box)
- Submit your assignment in Week1 DropBox in BrightSpace (in your Lab Section's DropBox)
- The due date for each assignment is on DropBox

LINUX
OPERATING SYSTEM I

(CST8207)

UNIX/LINUX

File Directories

Structure & manupilation

Algonquin College

Week2

By: Arsalan

LINUX/UNIX Desktop Options

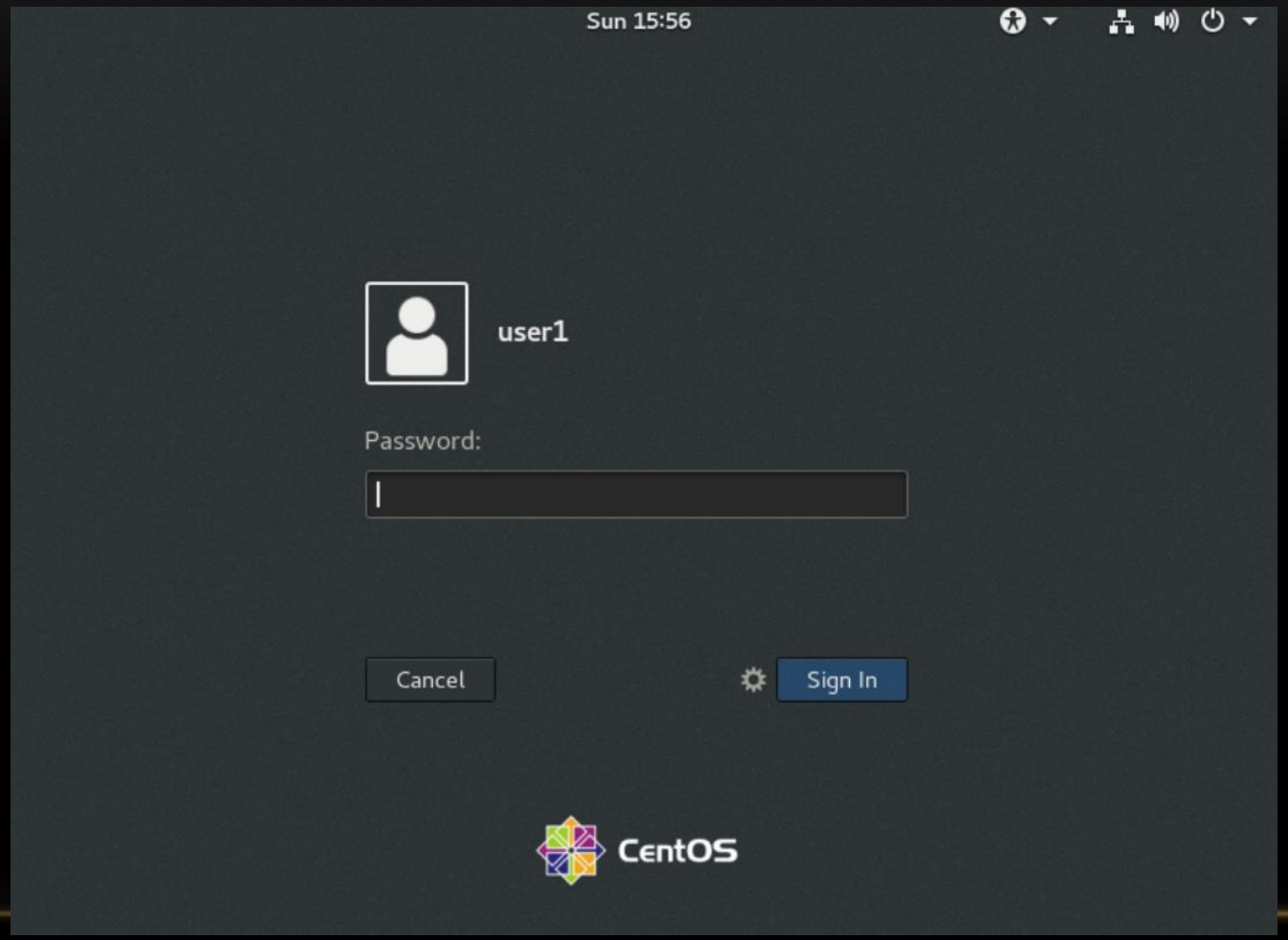
There are (2) main UNIX interfaces

- Command-line Shell Prompt (text-based ; CLI)

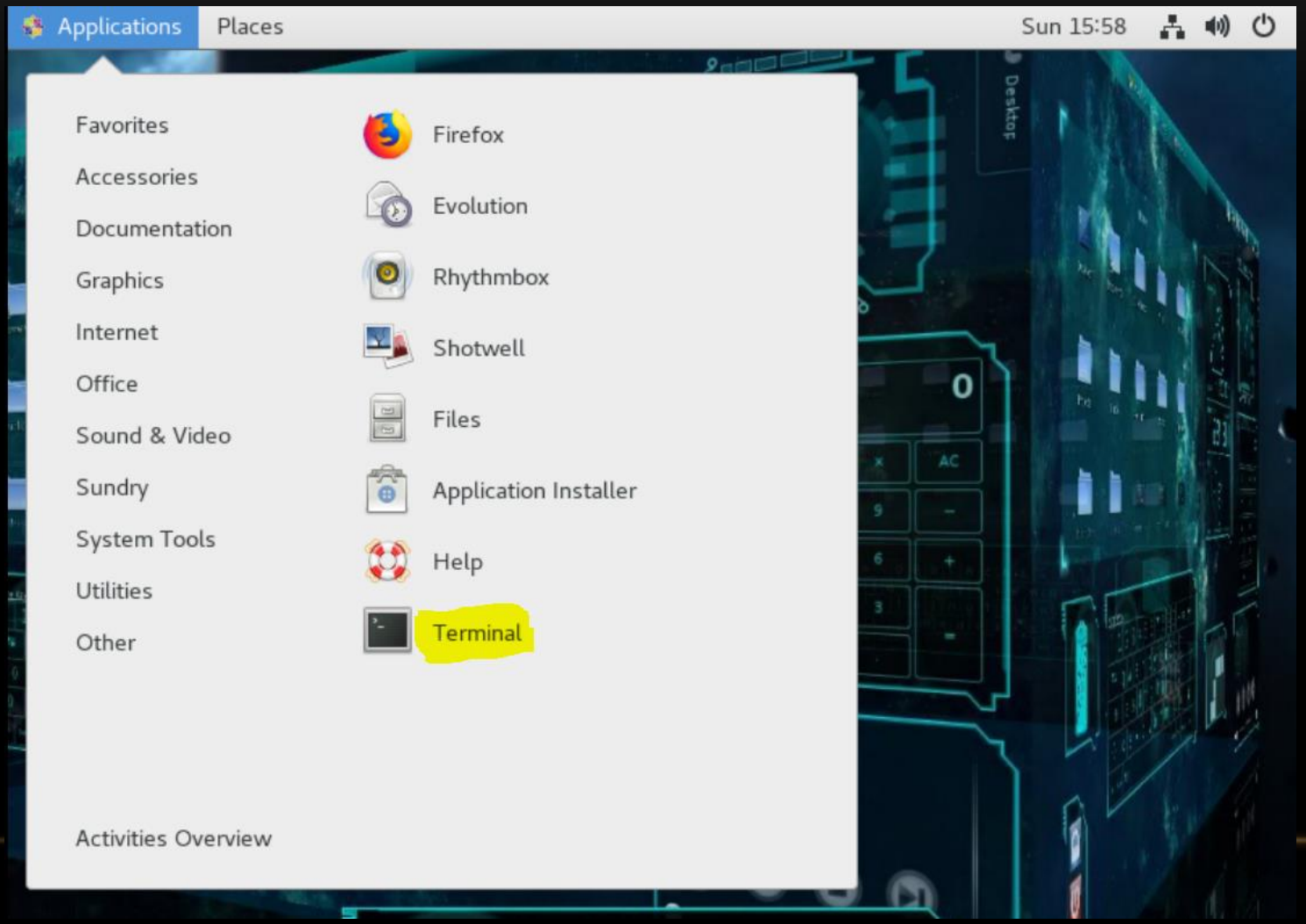
- Graphical Desktop (GUI)

The command-line provides more functionality, and supports script execution

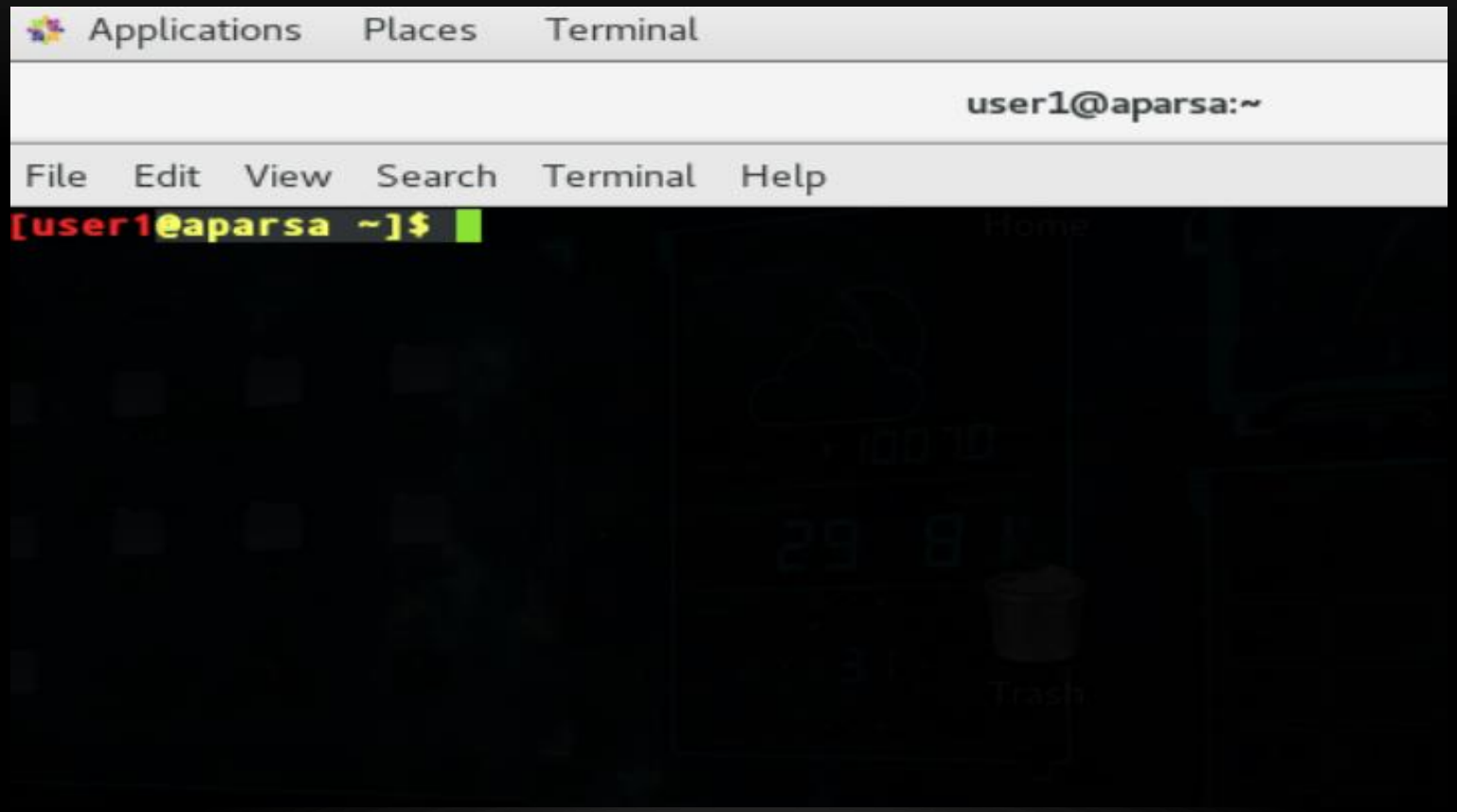
LOGIN SCREEN GUI



ACCESS TO TERMINAL



UNIX/LINUX TERMINAL (CLI) INTERFACE



What is Filesystem Directories ?

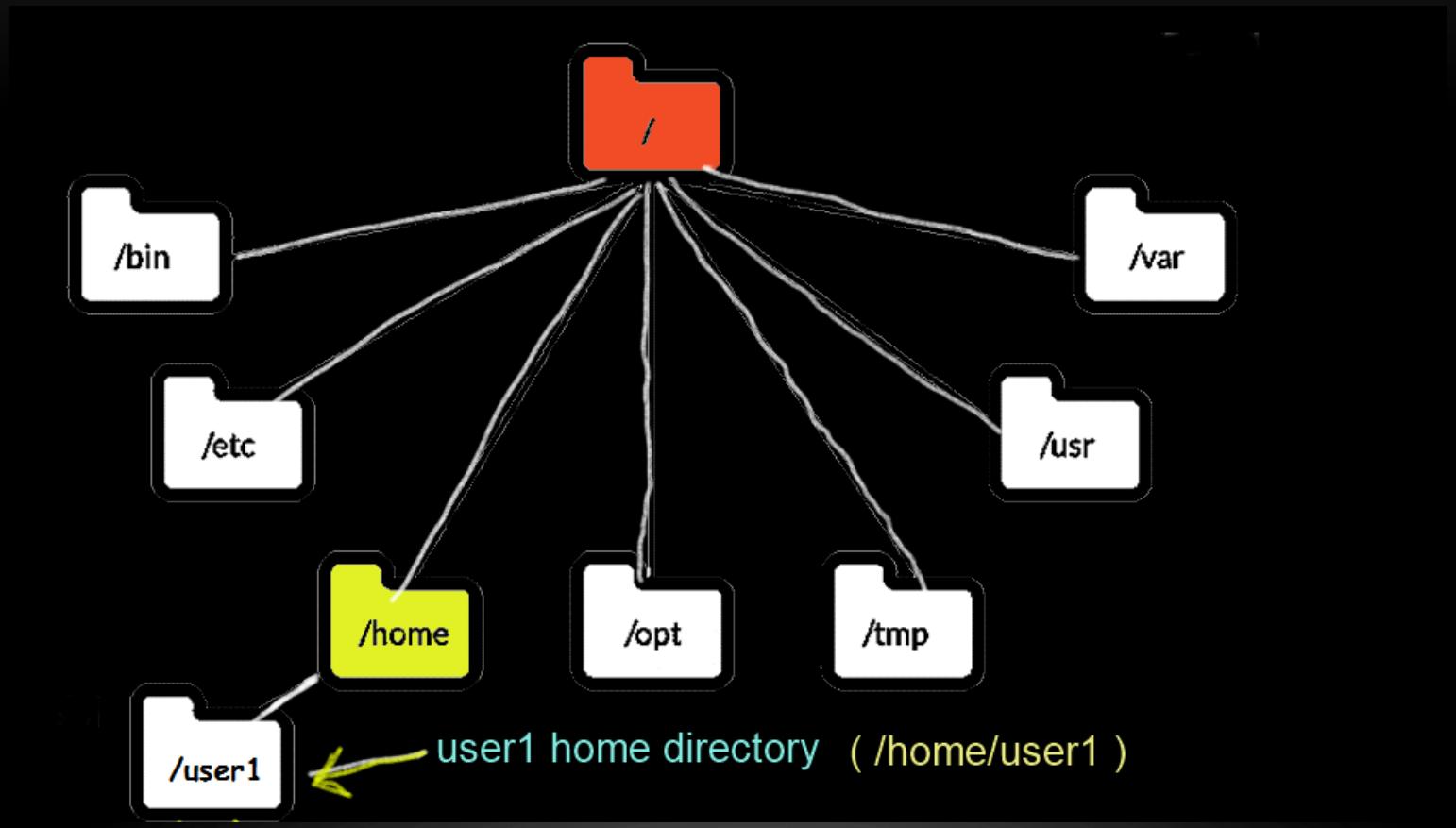
In LINUX/UNIX, all directories and files are stored in one common hierarchal tree structure with the **root directory** (designated as: ***/***) at the top of the structure

There is **no concept of volume letters** (e.g. C:, F:, K:, etc.) as is commonly used in a Microsoft Windows operating system

Folder (GUI)

Directory (CLI)

UNIX/LINUX DIRECTORY STRUCTURE



Home Directory

When a user login and access CLI terminal , they are placed in their user account's **home directory.**

It is special and it creates when a user account is created

Typically each user has his/her own home directory where the user stores personal files

This keeps a user's files separate from the files of other users who may sign on to the same node

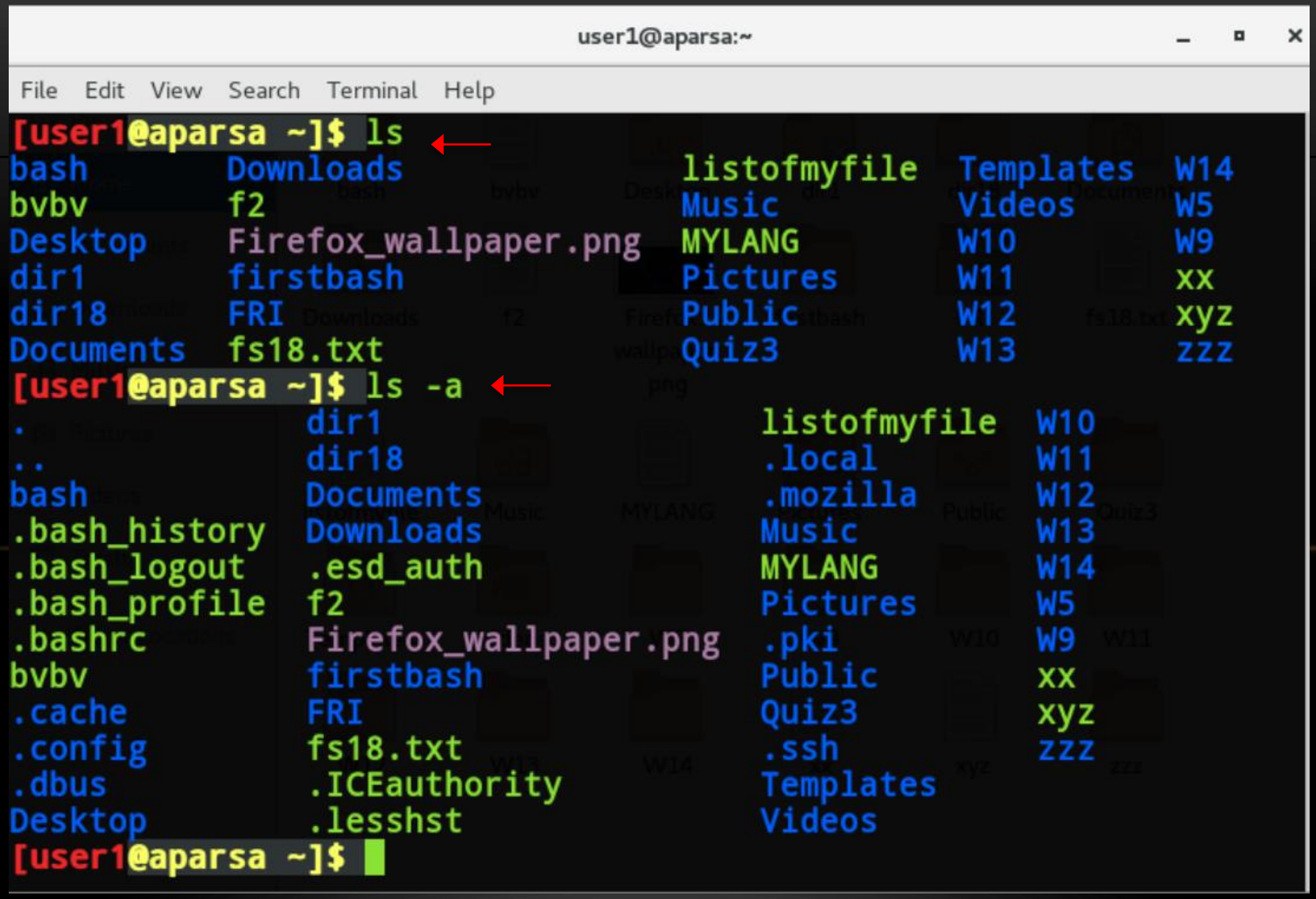
A user's home directory is designated as (~)

Utility: ls

Usage:

$ **ls** (or) **ls -a -l** (or) $ **ls -al**

* List all files and directories (including the hidden ones) located at my current directory context. Provide detailed information about each file and directory


user1@aparsa:~
File Edit View Search Terminal Help
[user1@aparsa ~]$ ls
bash Downloads listofmyfile Templates W14
bvbv f2 Music Videos W5
Desktop Firefox_wallpaper.png MYLANG W10 W9
dir1 firstbash Pictures W11 xx
dir18 FRI Public W12 xyz
Documents fs18.txt Quiz3 W13 zzz
[user1@aparsa ~]$ ls -a
. dir1 listofmyfile W10
.. dir18 .local W11
bash Documents .mozilla W12
.bash_history Downloads Music W13
.bash_logout .esd_auth MYLANG W14
.bash_profile f2 Pictures W5
.bashrc Firefox_wallpaper.png .pki W9
bvbv firstbash Public xx
.cache FRI Quiz3 xyz
.config fs18.txt .ssh zzz
.dbus .ICEauthority Templates
Desktop .lesshst Videos
[user1@aparsa ~]$

Utility: cd

The (**cd**) utility is used to change your current directory context

Usage:

$ **cd**
  * Change my directory context to my home directory

$ **cd ~**
  * Change my directory context to my home directory

$ **cd Downloads**
  * Change my directory context to the (Downloads) directory, which is located immediately below my current directory context

Utility: pwd

The (**pwd**) utility is used to identify your current directory context on the filesystem (which directory you are currently in)

Usage:

```
$ pwd
```

FIRST SKILL ABOUT WORKING WITH COMMAND-LINE (CLI)

- Know in which directory of file-system-tree you are sitting?
- Know what command tells you what directory you are in?
- Know how traverse among different directories

REVIEW QUESTIONS WEEK2

What are the (4) of the major components in all UNIX and UNIX “like” operating systems versions?

- What is home directory ?
- What is pwd command for?
- What is cd command for?
- What / means in UNIX/LINUX file system?
- What is the Job of Kernel?
- What is the job of Shell?

WHAT TO DO THIS WEEK?

- File/Directory to be continued in next lecture
- Read and complete assignment#2 for this week labs
- Read the Week 2 contents and online resource
- As the weekly lectures are split in two, We will continue this topic on next lecture to complete it
- In Assignment 2, you will find more commands and example of how to use it
  - Don’t forget Set up your Linux Box in the Lab (Assignment 1) and submit

LINUX
OPERATING SYSTEM I

(CST8207)

UNIX/LINUX

Globbing

I/O Redirection

Pipe Lining

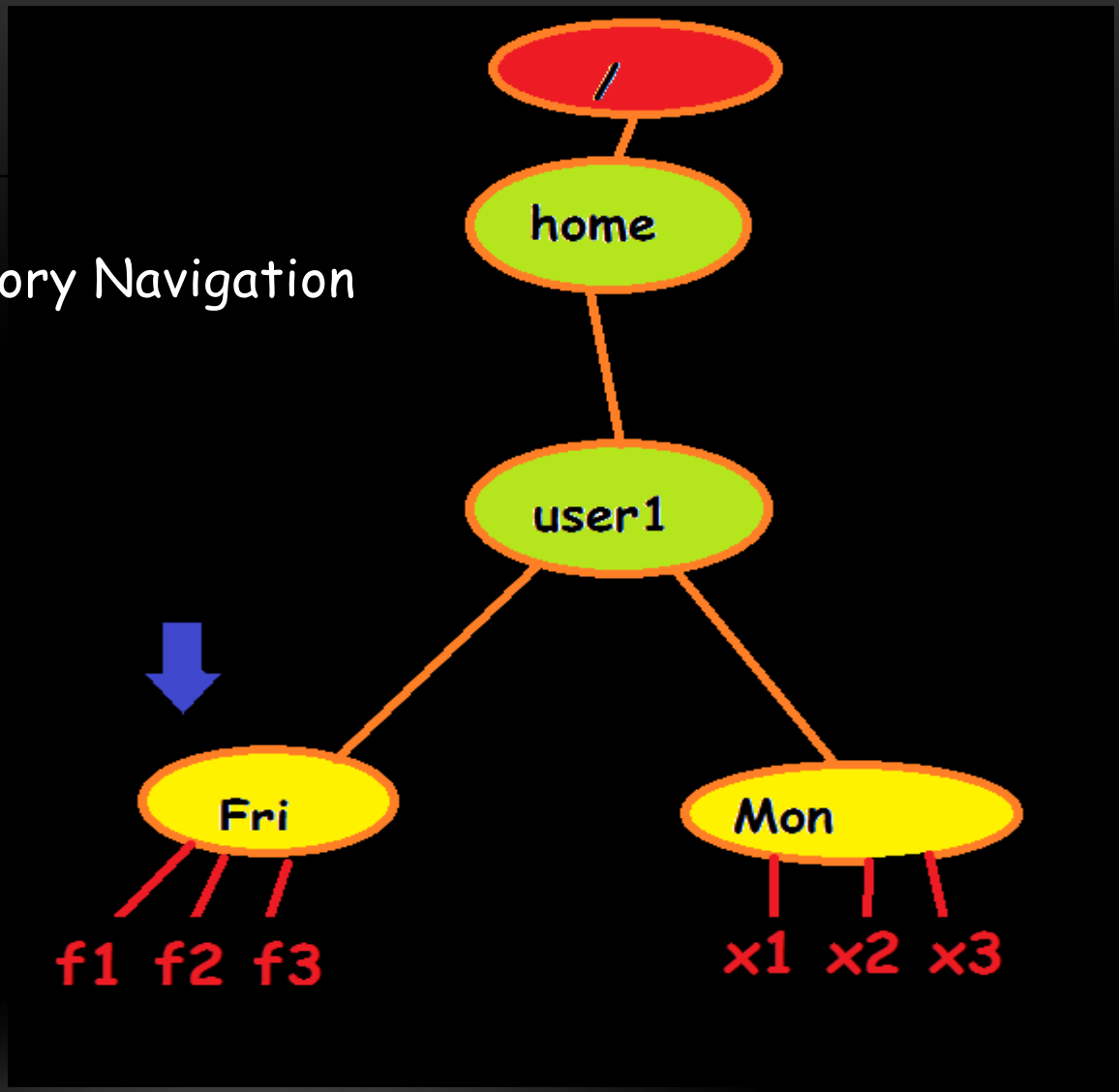
Algonquin College

Week3

By: Arsalan Parsaei

PREVIOUSLY …

- Linux File/Directory Navigation
- Absolute Path
- Relative Path



Note to remember:

- In Unix/Unix_Like OS, All directories are located below the root (/)
- All directories are separated by the use of a slash character (**/**)
- All commands, file and directory's names are **case sensitive**
- File/Directory Path:
  1) **Absolute Path**
  2) **Relative Path** (to your directory you are in)

LINUX GLOBBING (PATHNAME EXPANSION)

- For faster and effective file manipulation we use Globbing. There are some special characters to use as a wildcard for pattern matching in file or directory name

Char	What does it match with
*	zero (0) or more characters
?	any one (1) character (including blanks, punctuation, and unprintable characters!)
[aeAd]	exactly one (1) character in the list between brackets: a, e, A, or d
[!abc]	exactly one (1) character that is not in the list of characters (sometimes written as [^abc])

e.g: look at below examples:

LINUX PIPING

- We could send data from one program to another. It's called piping and the operator we use is (|)
- The output from one program become the input for another program
- e.g: $ ls / | more (sending output of ls to more command which shows out put page by page
- **$ ls / | sort | head -5** (send output of ls / to the sort command and show 5 first lines
- **$ ls /home/user1 | wc -l** (send out of ls command to wc to count how many line the output are)

I/O REDIRECTION

- Input and output in the Linux environment is distributed across three streams

Standard Input (stdin) symbol "<" e.g: $ cat < file1.txt

Standard Output (stdout) symbol ">" e.g: $ ls /etc/passwd > file1.txt

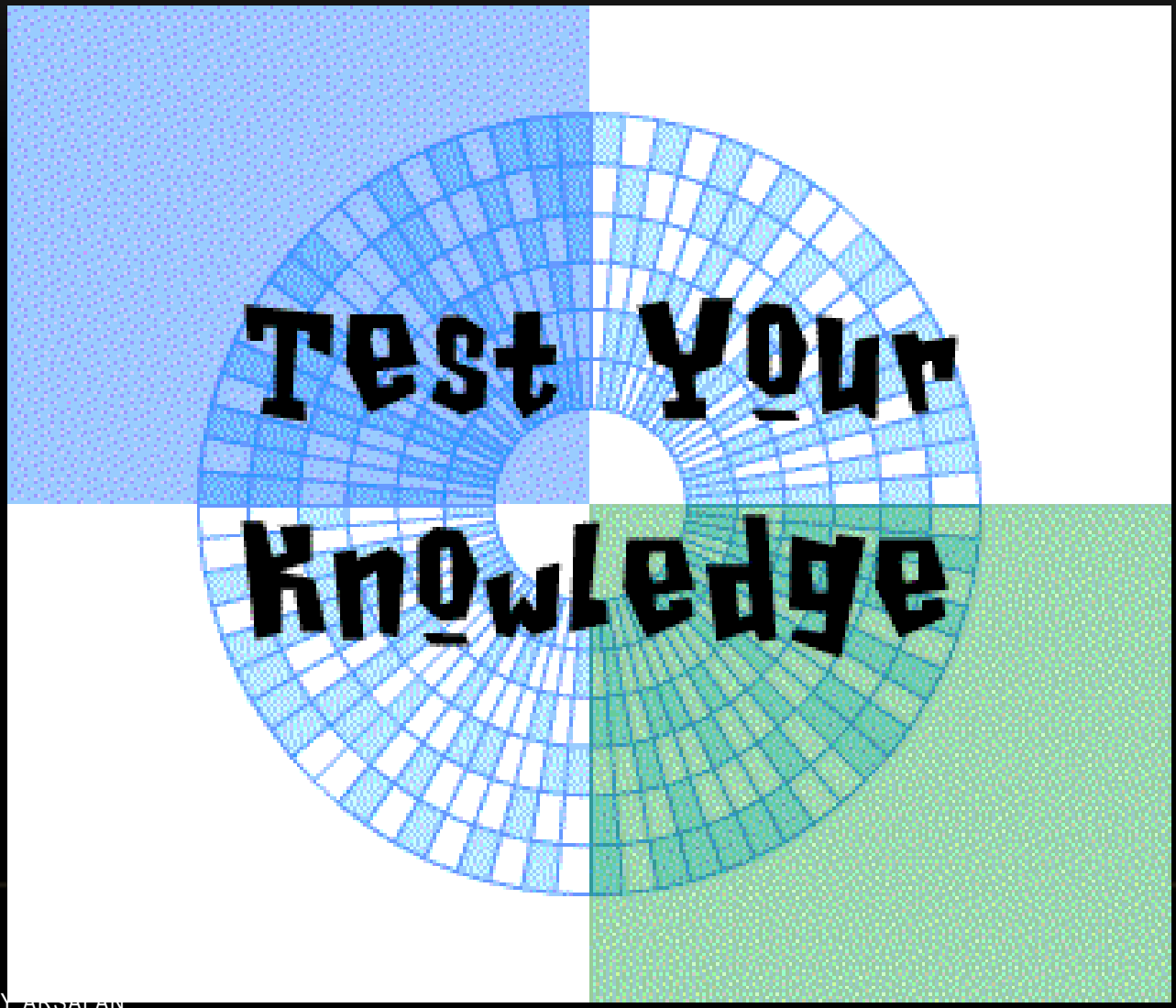
Standard Error (stderr) symbol "2>" e.g: $ mkdir dir1 di2 2> error.log

- The standard input (stdin) device is reading from the keyboard but you could redirect it to read from a file
- The standard output (stdout) device is the screen but your could redirect it to send output to a file
- The standard error, by default, error stream is displayed on Terminal. Error redirection is routing the errors to a file other than the screen.
- ">" is the output redirection operator. ">>" appends output to an existing file
- Tip: **2>&1** send all error message where the standard output goes

NOTE TO REMEMBER

- Pipes '|' send the output of one command as input of another command.
- The Filter takes input from one command, does some processing, and gives output.
- The grep command can be used to find strings and values in a text document
- Piping through grep has to be one of the most common uses
- 'sort' command sorts out the content of a file alphabetically
- less ,more commands are used for dividing a long file into page by page view
- Command “wc” is used for counting # of lines, words, bytes in a file
- Command “grep” is used for searching a pattern in a given file or stream

CHALLENGE



IN-CLASS PRACTICE

- Create a new empty file call f1.txt at my current directory context
- Change my directory context to my home directory
- Redirect the output of ls command in to file1.txt
- Redirect the possible error messages into file “error.log”

- Change my directory context to the (Videos) directory, which is located (1) directory above my current directory context
- Change my directory context to the (Videos) directory, which is located (3) directories above my current directory context

WHAT TO DO FOR NEXT CLASS

- Complete assignment#3 in this week lab
- Read the Week 3 contents and resources and practice
- Practice “just for practice” in Week3 contents I have created for you
- Participate in Discussion board

LINUX
OPERATING SYSTEM I

(CST8207)

UNIX/LINUX
Search/Permission/Aliasing
File/Dir Details

Algonquin College

Week4

By: Arsalan Parsaei

File and Directory Details

When inspecting file or directory details at a command-line level, at lot of information will be provided. For example:

```
[user1@aparsaei W20]$ ls
Dir1  Dir2  f1  f2  f3.txt  f4.docx  f5  f6
[user1@aparsaei W20]$ ls -l
total 0
drwxrwxr-x. 2 user1 user1 6 Jan 14 11:45 Dir1
drwxrwxr-x. 2 user1 user1 6 Jan 14 11:45 Dir2
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f1
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f2
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f3.txt
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f4.docx
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f5
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f6
[user1@aparsaei W20]$
```

File and Directory Details

-rw-r-xr-w. 1 user1 user1 4096 Mar 2 13:30 f1.docx
drwxrwxr-w. 4 user1 user1 37 Jan 20 12:20 f2.txt

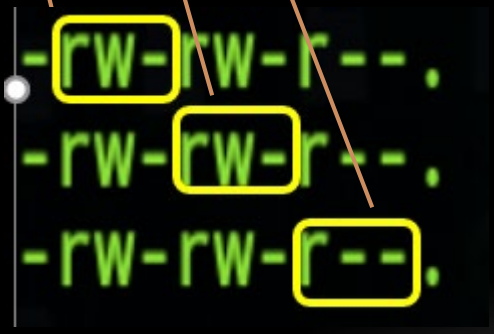
d : Indicates the file is a directory

- : Indicates the file is a regular file

```
[user1@aparsaei W20]$ ls
Dir1  Dir2  f1  f2  f3.txt
[user1@aparsaei W20]$ ls -l
total 0
drwxrwxr-x. 2 user1 user1 6
drwxrwxr-x. 2 user1 user1 6
-rw-rw-r--. 1 user1 user1 0
-rw-rw-r--. 1 user1 user1 0
-rw-rw-r--. 1 user1 user1 0
-rw-rw-r--. 1 user1 user1 0
-rw-rw-r--. 1 user1 user1 0
-rw-rw-r--. 1 user1 user1 0
[user1@aparsaei W20]$
```

FILE/DIRECTORY PERMISSION

- **Owner permissions** – what actions the owner of the file can do on the file.
- **Group permissions** – what actions a user's group can do on the file.
- **Other permissions** – what action all everyone can do on the file.




```
[user1@aparsaei W20]$ ls
Dir1  Dir2  f1  f2  f3.txt  f4.docx  f5  f6
[user1@aparsaei W20]$ ls -l
total 0
drwxrwxr-x. 2 user1 user1 6 Jan 14 11:45 Dir1
drwxrwxr-x. 2 user1 user1 6 Jan 14 11:45 Dir2
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f1
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f2
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f3.txt
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f4.docx
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f5
-rw-rw-r--. 1 user1 user1 0 Jan 14 11:45 f6
[user1@aparsaei W20]$
```

UNIX/LINUX FILE/DIRECTORY PERMISSION

- First line of the defense in term of the security of a Unix system.
- How to control file/directory Permissions?
- **File** Access Mode: **Read Write Execute**

- **Directory** Access Mode: **Read Write Execute**
- Execute: Oh what could that possibly mean for a directory… ??? Traverse, if you don't have the permission you could change to that directory (cd)

ABSOLUTE MODE / SYMBOLIC MODE

Symbol		Octal	Decimal
---	No rwx	0 0 0	0
--x	Only execute	0 0 1	1
-w-	Only write	0 1 0	2
-wx	Write & Exec	0 1 1	3
r--	Only read	1 0 0	4
r-x	Read & Exec	1 0 1	5
rw-	Read & Write	1 1 0	6
rwx	Full permission	1 1 1	7


[user1@aparsaei W20]$ getfacl Dir1
file: Dir1
owner: user1
group: user1
user::-w-
group::rw-
other::rw-
[user1@aparsaei W20]$ getfacl f1
file: f1
owner: user1
group: user1
user::rw-
group::rw-
other::r--

No Execute

```
[user1@aparsaei W20]$ chmod 666 Dir1
[user1@aparsaei W20]$ cd Dir1
bash: cd: Dir1: Permission denied
[user1@aparsaei W20]$ ls Dir1
ls: cannot access Dir1/x1: Permission denied
x1
```

No Read

```
[user1@aparsaei W20]$ chmod 266 Dir1
[user1@aparsaei W20]$ ls Dir1
ls: cannot open directory Dir1: Permission denied
[user1@aparsaei W20]$ cd Dir1
bash: cd: Dir1: Permission denied
```


[user1@aparsaei W20]$ getfacl Dir1
file: Dir1
owner: user1
group: user1
user::-w-
group::rw-
other::rw-
[user1@aparsaei W20]$ touch Dir1/x2
touch: cannot touch 'Dir1/x2': Permission denied
[user1@aparsaei W20]$ cd Dir1
bash: cd: Dir1: Permission denied
[user1@aparsaei W20]$ ls Dir1
ls: cannot open directory Dir1: Permission denied
[user1@aparsaei W20]$

TWO TIPS AND HELPFUL UTILITIES

- **Find**
- **Alias**

$ find

- The (**find**) : locate files and directories on the filesystem.
- The (**find**) utility is one of the most important utility to locate stuff on drive

How to deploy find:

```
$ find / -type f -name "*.txt"
```

* Locate all files ending with (.txt) on the entire filesystems

```
$ find / -type f -iname "f1.txt" 2> /dev/null
```

* Locate all files called (f1.txt) on the entire filesystems, and trash out any error message on screen

```
$ find / -type d -iname "*file*" 2> /dev/null
```

* Locate all directories with the string ("file") in their name

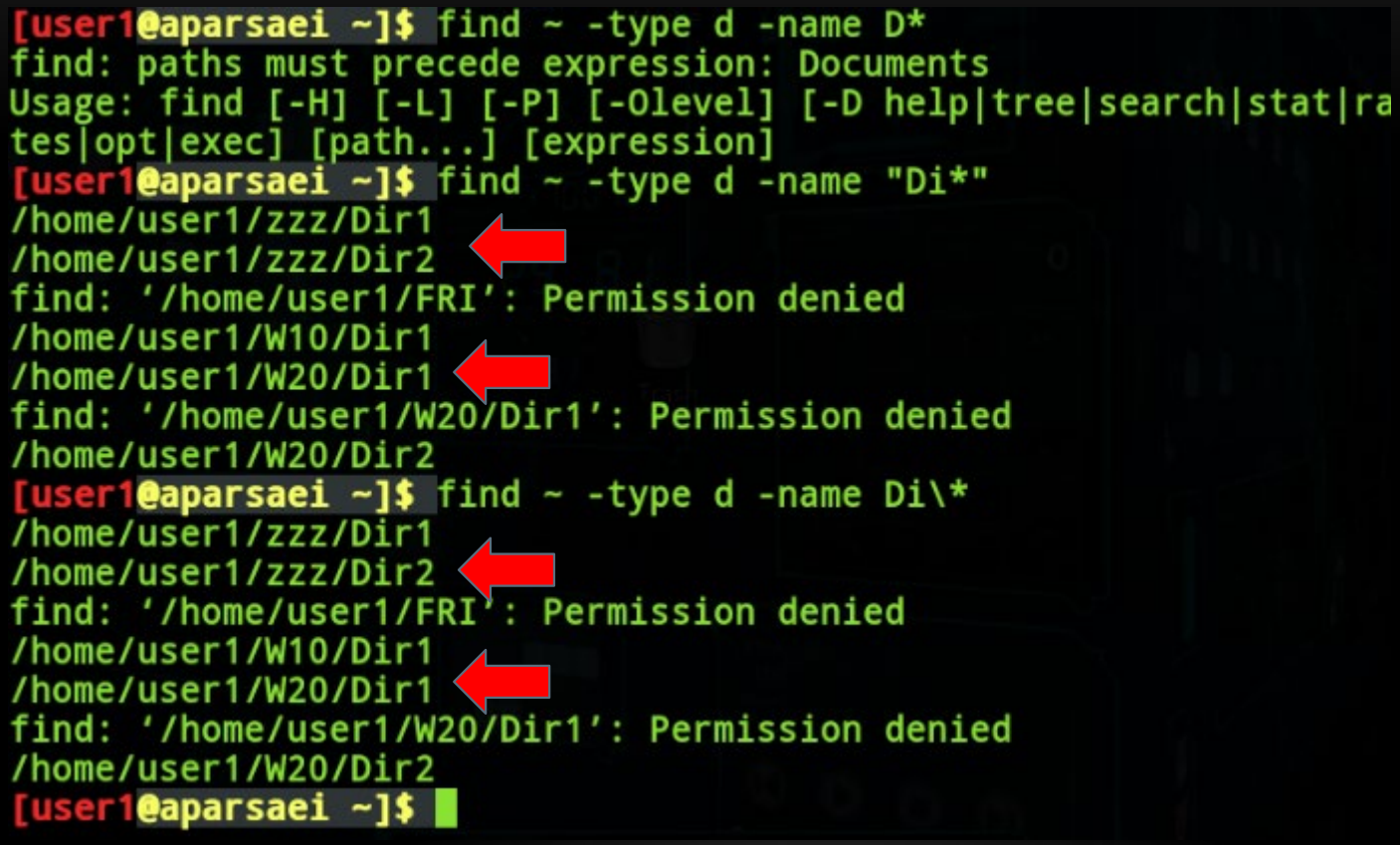
-type f : Locates only files

-type d : Locates only directories

-iname or name : make it case **insensitive**

2> /dev/null : Do not display any access error messages on screen

POSSIBLE ERROR IN FIND COMMAND USE “” IN SEARCH PATTERN

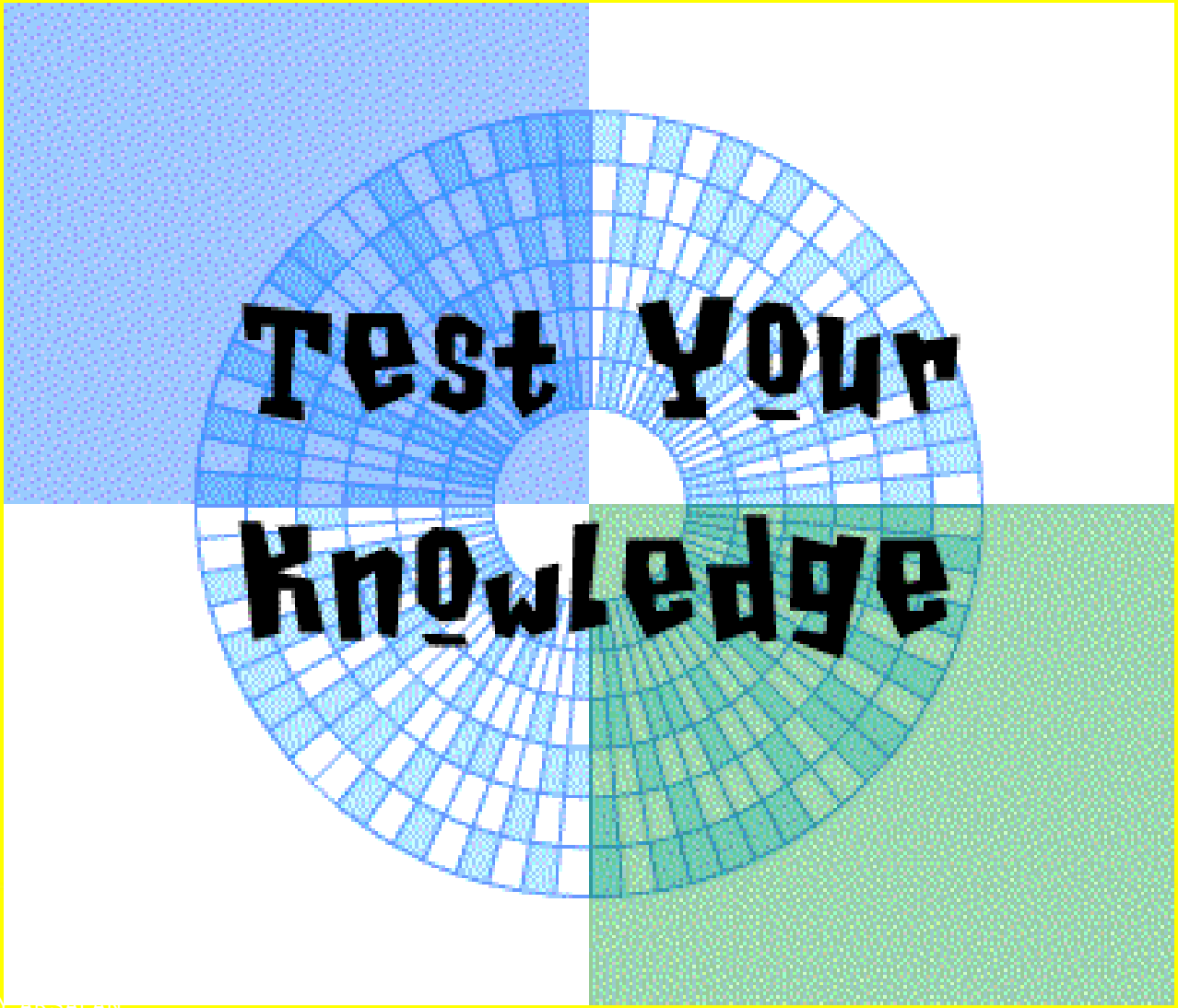


ALIAS, COMMAND CREATING

- **c: to clear screen $alias c='clear'**
- **b to get a long list, $ alias b='ls -al'**
- **Destroy : wipeout hard drive (hint: rm –rf)**
- **Off: turn off machine gracefully (hint: shutdown)**

- **To remove alias enter $ unalias c**

CHALLENGE



- List detailed of all the non-hidden files and directories located at my current directory context
- List all files and directories (including the hidden ones) located at my current directory context
- Create a new empty file at my current directory context
- Change my directory context to my home directory
- Redirect the output of ls command in to file1.txt
- Redirect the possible error messages into file “error.log”
- Write an Alias called “Bday” such that it shows your birthday when you run it

- Change my directory context to the (Videos) directory, which is located (1) directory above my current directory context
- Change my directory context to the (Videos) directory, which is located (3) directories above my current directory context
- Count the # of line in weekdays file
- Display # of words in the file “weekdays”
- Match file names which start with letter “f”
- Match file names which start with letter “f” and have 3 letters in their name

WHAT TO DO FOR NEXT CLASS

- Complete assignment#4 in this week lab
- Read the Week 4 contents and resources and practice
- Practice “just for practice” in Week4 contents I have created for you
- Finish Online short Quiz1

LINUX
SYSTEM SUPPORT

(CST8207)

Linux Variables
User Start-up Script
Shell Parameter Expansion

Algonquin College

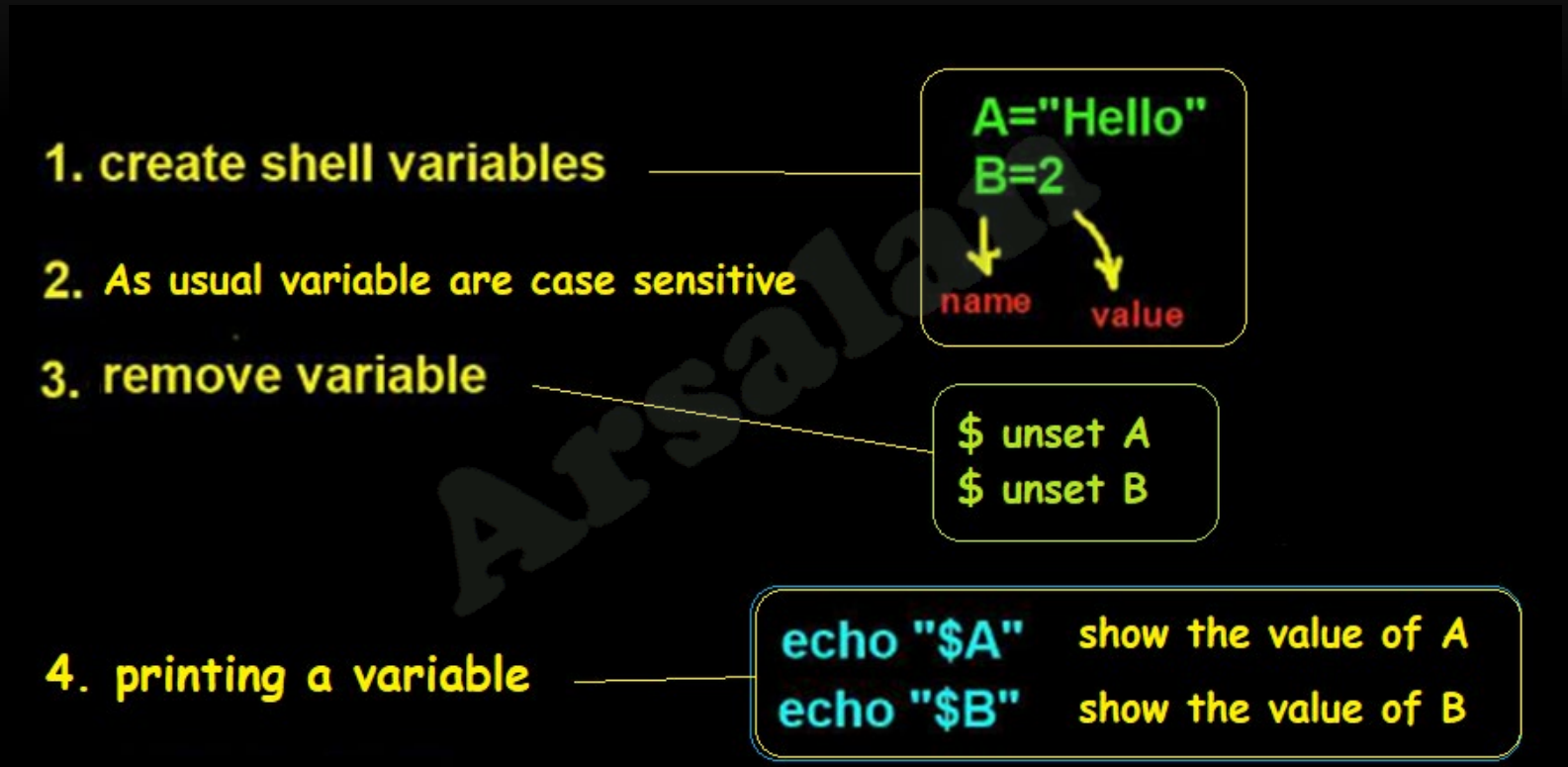
Week5

By: Arsalan Parsaei

WEEK 5

- Thursday Lecture to review Quiz 1
- Lecture 5
- Test Your Knowledge in class challenge
- Assignment #5

WORKING WITH SHELL VARIABLES



CREATING SAMPLE SCRIPT IN BASH

- Variable names : include only (a to z or A to Z),(0 to 9) or (_)
- By convention, Linux/UNIX shell variables will have their names in UPPERCASE.
- To retrieve the value of a variable in script we use **echo** command

```
#!/bin/bash
NAME="Bill Gates"
echo $NAME
```

Create a file called "name.sh" and using your editor , write this code in the file and save it .

Then give executable permission to using:
$chmod 755 name.sh
Then execute:
$ ./name.sh

This called a bash script

UNSETTING VARIABLE & STARTUP SCRIPTS

Unsetting variables in script

```
#!/bin/bash
NAME="Bill Gates"
unset NAME
echo $NAME
```

Create a file called "name2.sh" and using your editor , write this code in the file and save it .

Then give executable permission to using:
$chmod 755 name2.sh

Then execute:
$ ./name2.sh

Because the variable is undet your don't see the name in the output this time

There are different ways to make startup script that runs anytime you start you Linux box. In the future you learn more technique about script startup & automation
At this stage, remember, you can use same practice (script inside .bashrc) to run when you open shell, or ~/.bash.logout to be executed when you log out of the shell.(using exit or logout command)

Live Demo

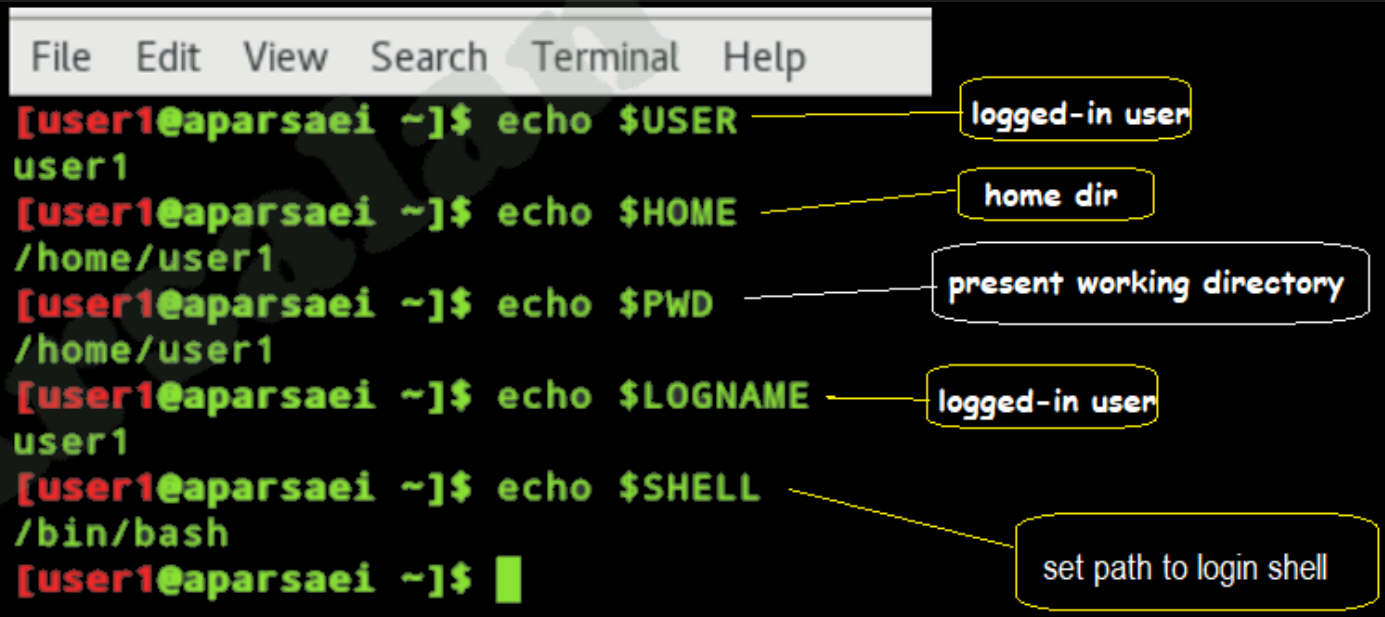
Shell Expansion

Writing the first simple script

Variables in Linux

Set , Unset Variables

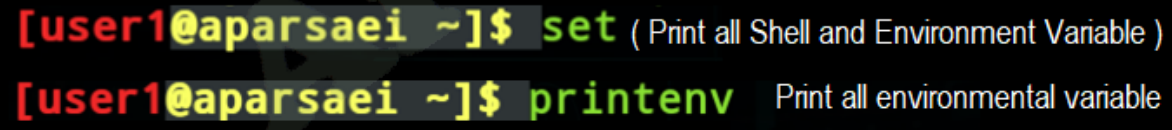
ENVIRONMENTAL VARIABLE



In future, we will discuss more about the env. variable and difference

TRY THEM:

- Create variable ANSWER with value of 2
  - ANSWER = 2
- Copy the value of ANSWER to variable CORRECT
  - CORRECT = $ANSWER
- Destroy variable ANSWER -> $ unset ANSWER
- List all the variables:



SHELL/ENV VARIABLES

- set – The command sets or unsets shell variables. When used without an argument it will print a list of all variables including environment and shell variables, and shell functions.
- unset – The command deletes shell and environment variables.
- export – make the variables global (available out of current shell or to child processes)

```
[user1@aparsaei W5]$ printenv LANG PWD HOME
en_CA.UTF-8
/home/user1/W5
/home/user1
[user1@aparsaei W5]$
[user1@aparsaei W5]$ echo $BASH_VERSION
4.2.46(2)-release
```

Analyzing the Effect of exporting variable

To make the environmental variable available to other child process or functions, we use export command. “export” is basically a built-in shell command for Bash (used to export an environment variable to be inherited by child process. try this too understand the effect:

```
[user1@aparsaei ~]$ ZZ=2
[user1@aparsaei ~]$ set | grep ZZ
ZZ=2
[user1@aparsaei ~]$ env | grep ZZ
[user1@aparsaei ~]$
[user1@aparsaei ~]$ export ZZ=2
[user1@aparsaei ~]$ env | grep ZZ
ZZ=2
```

“set” VS “printenv” command:

Both this commands show the variables already exists. However, “set” sees the local variable (it is built-in command) but printenv doesn’t as it is an independent executable. As an example to see the effect, I created a variable ZZ=2, this is a local variable which is created, to make it Environment variables, I used export ZZ=2

Live Demo

Variables in Linux

Prompt manipulation

Disk usage

SHELL PROMPT MANIPULATION

- Prompt in Linux is stored in **PS1** Environmental variable:
- $ echo **$PS1**

```
[user1@aparsaei ~]$ PS1='\u:\w\$'
user1:~$
```

- PS1="[\u@\h \W \@]\$"
- Change prompt color:
  - PS1='\e[0;31m[\u@\h \W]\$ \e[m'

-
```
[user1@aparsaei ~]$ PS1='\t \u@\h $ '
17:56:44 user1@aparsaei $
17:56:53 user1@aparsaei $
```

More of the bash prompt and colors -> (Prompt manipulation)

MAKE YOU PROMPT PERMANENT

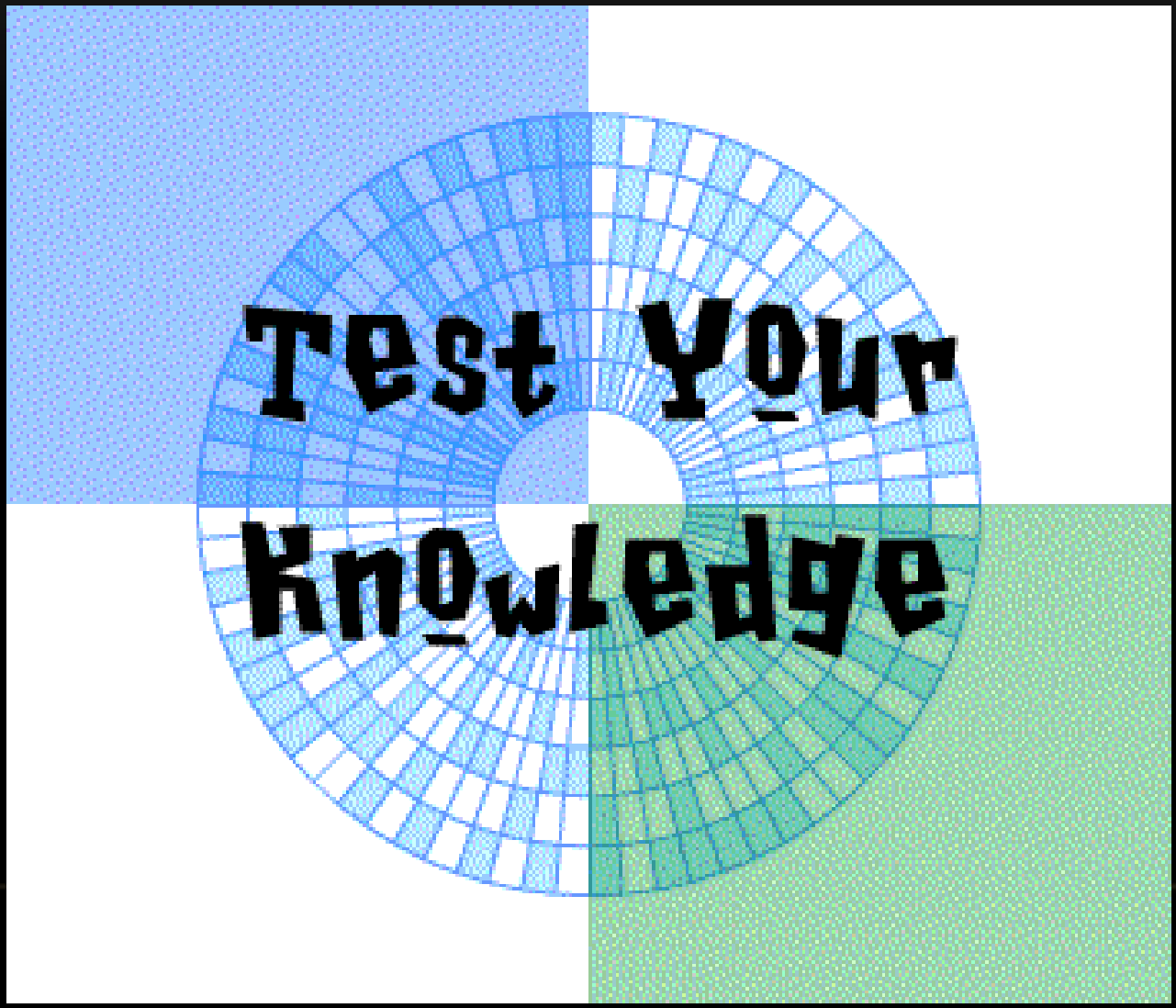
- Don’t forget to do two things when you have finalized you prompt:
  - 1) use export for permanent changes
    - `export PS1="[\u@\h \W \@]\$"`
  - 2) To avoid temporary prompt, add your PS1 variable in your ~/.bashrc

DISK USAGE:

- When it comes to file and directory administration, it is désirable to know the volume of those files/Directories for better files manipulation. One of the useful command is :
- du : it estimates file space usage on disk, basically it summarize disk usage of each file, and recursively for directories.
- There is different options for this command, some important one are
- du -a : write counts for all files, not just directories
- du -c : produce a grand total
- du –h : print sizes in human readable format (e.g., 1K 234M 2G)

Below example shows detail info about file/dirs. Disk usage. In /etc

CHALLENGE



- Create a variable MYNAME and make it permanent variable in your shell
- Find a way to add time to your Prompt
- List detailed of all the non-hidden files and directories located in /etc and save it to file ETC.TXT
- Display the number of files and directories in your home directory

WHAT TO DO FOR NEXT WEEK

- Complete assignment#5 in this week lab
- Read the Week 5 contents and resources and practice
- Set up your new prompt (PS1) as instructed in Assignment#5
- Lab Professors are to verify that you customized your Prompt
- Study for Midterm-TEST1 (Week 6)

LINUX
SYSTEM SUPPORT

(CST8207)

Managing Linux Process

Secure Shell

SSH.SCP

Algonquin College

Week6

By: Arsalan

WHAT IS PROCESS

A process is a program in execution;

Whenever a command is issued in unix/linux, it creates/starts a new process.

Through an ID number unix/linux keeps account of the processes, is call PID

Each process in the system has a unique pid.

Note: no 2 process with the same pid exist in the system

```
[user1@aparsaei ~]$ ps -c
  PID CLS PRI TTY          TIME CMD
 5424 TS   19 pts/0    00:00:00 bash
 5766 TS   19 pts/0    00:00:00 bash
 5810 TS   19 pts/0    00:00:00 bash
 5878 TS   19 pts/0    00:00:00 ps
[user1@aparsaei ~]$
```

PROCESS - TWO FACES OF EXECUTION

processes in Linux: could be run in bg or fg

fg: Foreground processes (interactive processes) –
initialized and controlled through a terminal session (need user to start them not automatic)

bg: Background processes (non-interactive/automatic processes)
not connected to a terminal;
no need for user input.

```
[user1@aparsaei ~]$ ps -A | grep gedit
 5921 pts/0    00:00:00 gedit
[user1@aparsaei ~]$ jobs
[1]+  Running                 gedit &
[user1@aparsaei ~]$
```

Live Demo

PROCESS

EXECUTION

TYPE

HOW TO MANAGE PROCESSES (COMMANDS)

what command shows running process?

```
$ ps -A    ps -e    ps -ax
```

-A and -e are basically equal

how to find process PID?

```
$ pgrep "process name"
```

How to terminate a process ?

```
$ kill "Process ID"        kill -9 "PID"
```

how to send a process to background ?

using "&" in front of process name, like :

```
$ firefox &
```

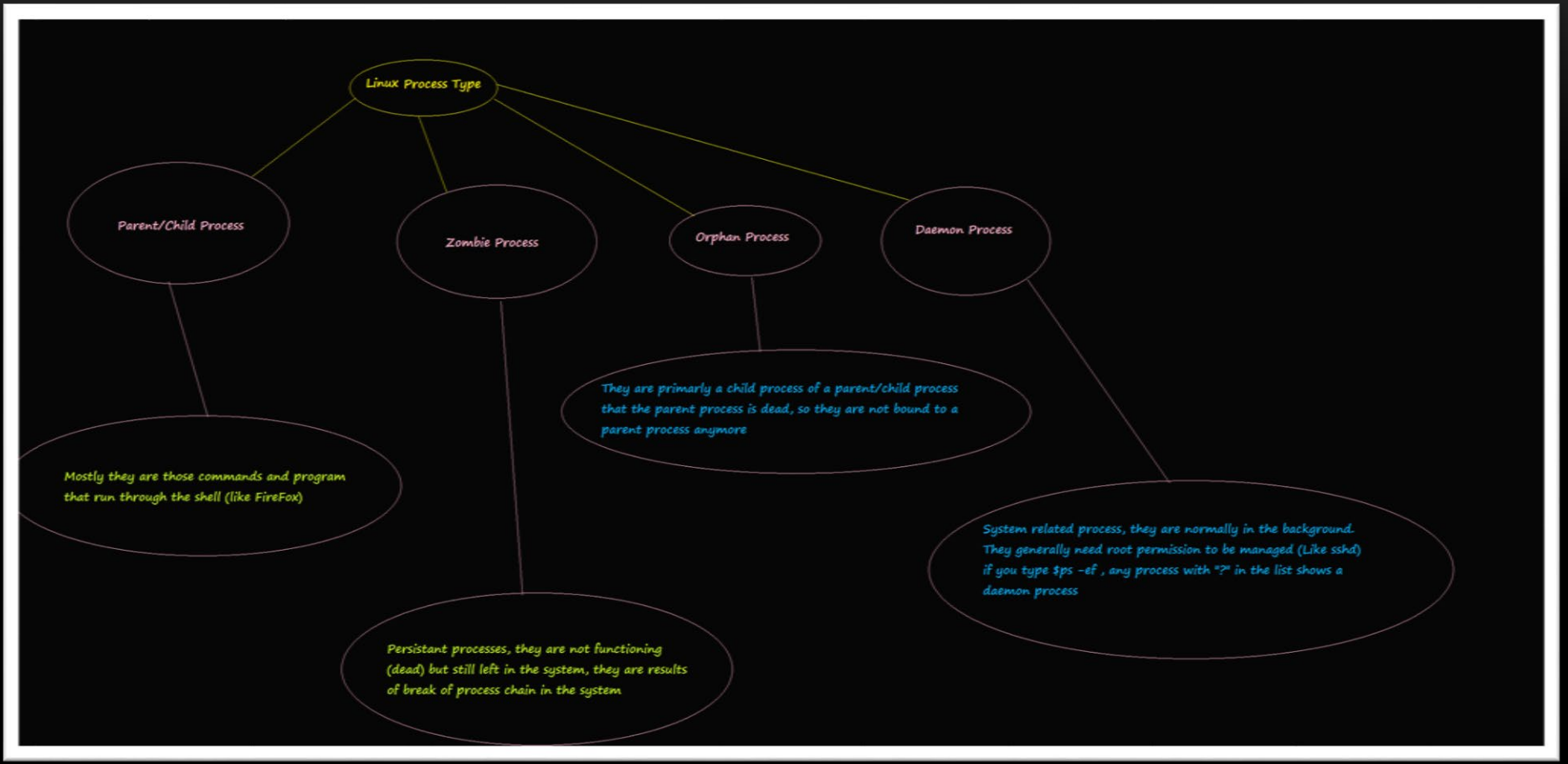
Live Demo

PROCESS MANAGEMENT

COMMAND & CONTROL

SEND PROCESS TO BACKGROUND
BRING PROCESS TO FOREGROUND

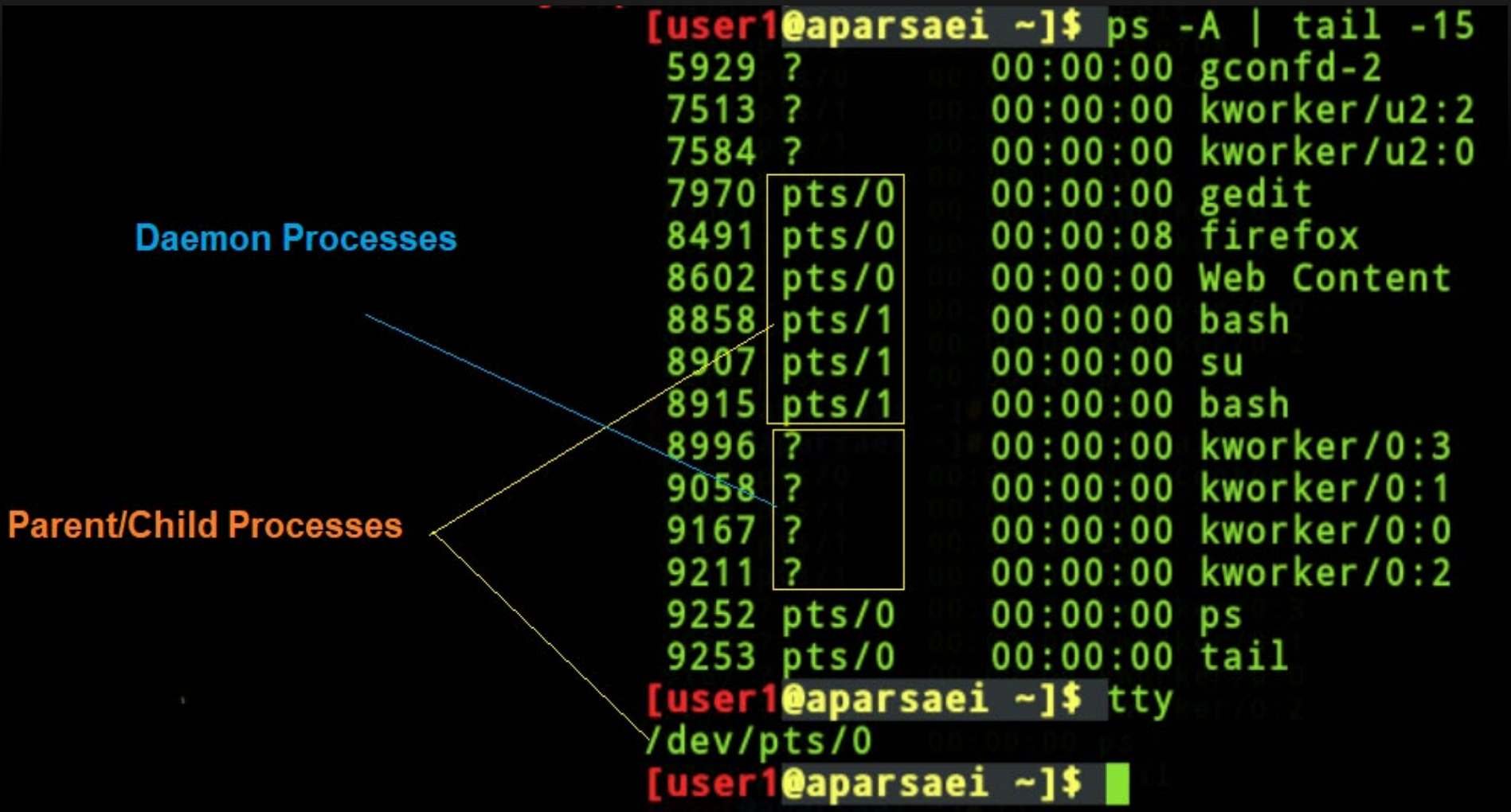
BELOW FIGURE SHOWS PROCESS TYPE



You could zoom in to see better, also The above firgure is up in your Week 6 content , you can download and see it full scale


```
[user1@aparsaei ~]$ gedit &
[1] 7651
[user1@aparsaei ~]$ firefox &
[2] 7664
[user1@aparsaei ~]$ jobs
[1]-  Running                 gedit &
[2]+  Running                 firefox &
[user1@aparsaei ~]$ pgrep gedit
7651
[user1@aparsaei ~]$ kill 7651
[1]-  Terminated              gedit
[user1@aparsaei ~]$ kill $(pgrep firefox)
[2]+  Terminated              firefox
[user1@aparsaei ~]$
```

EXAMPLE OF DIFFERENT PROCESS TYPE



Live Demo

PROCESS TYPES

PARENT/CHILD

O/Z

DAEMON

$ pstree is another command show the process tree and their parent process in a tree shape. Below example we have filtered output of pstree with bash

```
[user1@aparsaei ~]$ pstree | grep bash
        |-gnome-terminal--+-bash---bash---bash---bash---bash-+-grep
[user1@aparsaei ~]$
```

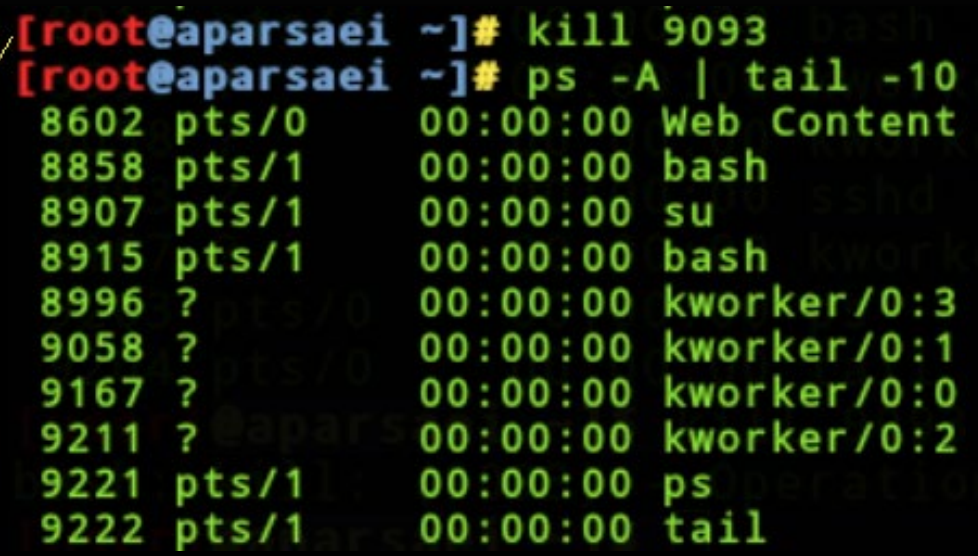
This tells us that we have 4 child process started from first bash shell terminal

```
[user1@aparsaei ~]$ pstree | grep bash
        |-gnome-terminal--+-bash---bash---bash---bash---bash-+-firefox-+-firefox
[user1@aparsaei ~]$
```

I ran firefox as bg process , compare with figure at the top, as you see above Firefox is linked to last bash (Parent) which tells me that bash terminal was the origin of Firefox (by command not from GUI)
In process analyze, this helps to find who is behind those malicious process of any kind. In windows we have similar process analyzer too, using tools like process explorer. In Linux you can use “top” command to serve you for that purpose.

```
[user1@aparsaei ~]$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
user1        537 32036  0 17:06 pts/0    00:00:00 ps -f
user1      22999 22991  0 Oct06 pts/0    00:00:00 bash
user1      23119 22999  0 Oct06 pts/0    00:00:00 bash
user1      31882 23119  0 16:50 pts/0    00:00:00 bash
user1      31992 31882  0 16:52 pts/0    00:00:00 bash
user1      32036 31992  0 16:52 pts/0    00:00:00 bash
user1      32131 32036 11 17:00 pts/0    00:00:36 /usr/lib64/firefox/firefox
user1      32523 32131  1 17:01 pts/0    00:00:04 /usr/lib64/firefox/firefox -contentproc
user1      32550 32131  0 17:01 pts/0    00:00:00 /usr/lib64/firefox/firefox -contentproc
user1      32599 32131  0 17:01 pts/0    00:00:01 /usr/lib64/firefox/firefox -contentproc
user1      32664 32131  0 17:01 pts/0    00:00:00 /usr/lib64/firefox/firefox -contentproc
[user1@aparsaei ~]$
```

to kill daemon you need root access



SECURE THE COMMUNICATION WITH ENCRYPTED CHANNEL (SSH)

- Secure communication is paramount in today world.
- It means that when you communicate through public network, you need to have some encryption protocol in place to make sure the information stays confidential.
- Linux has a secure channel suite called OpenSSH that provided secure login, secure communication as well as secure file copy from one host to another
- On your Linux Box you have OpenSSH suite. This suite has a daemon name sshd that run as a background process and handle remote login securely.
- It can establish encrypted channel between two hosts, and provide authentication/Encryption.

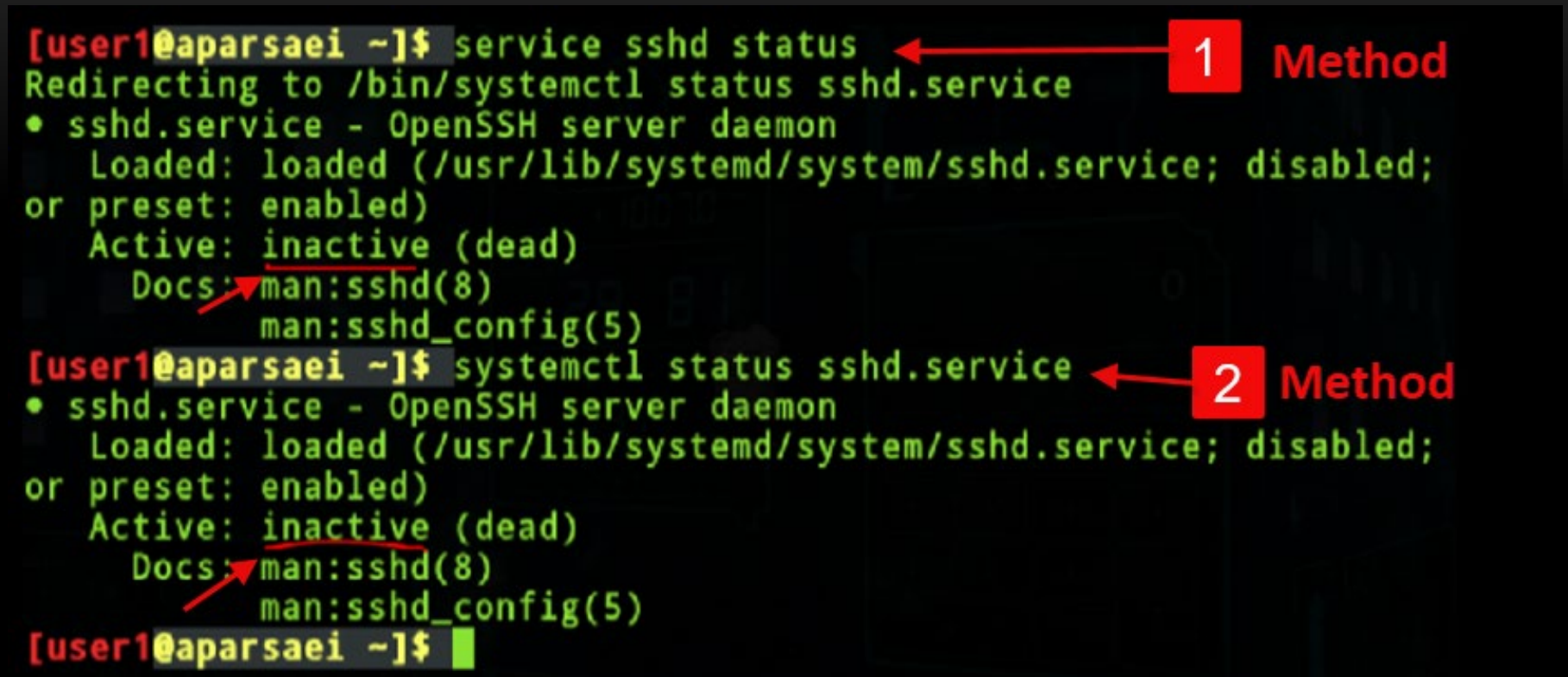
SSH DAEMON

- Secure sshd daemon
- Connect two remote hosts (Computers) together through Internet or Intranet
- Very secure and popular
- TOR – Stunnel
- To start/stop (use one of the following methods) (systemctl is newer) (Example of each method in the next slide

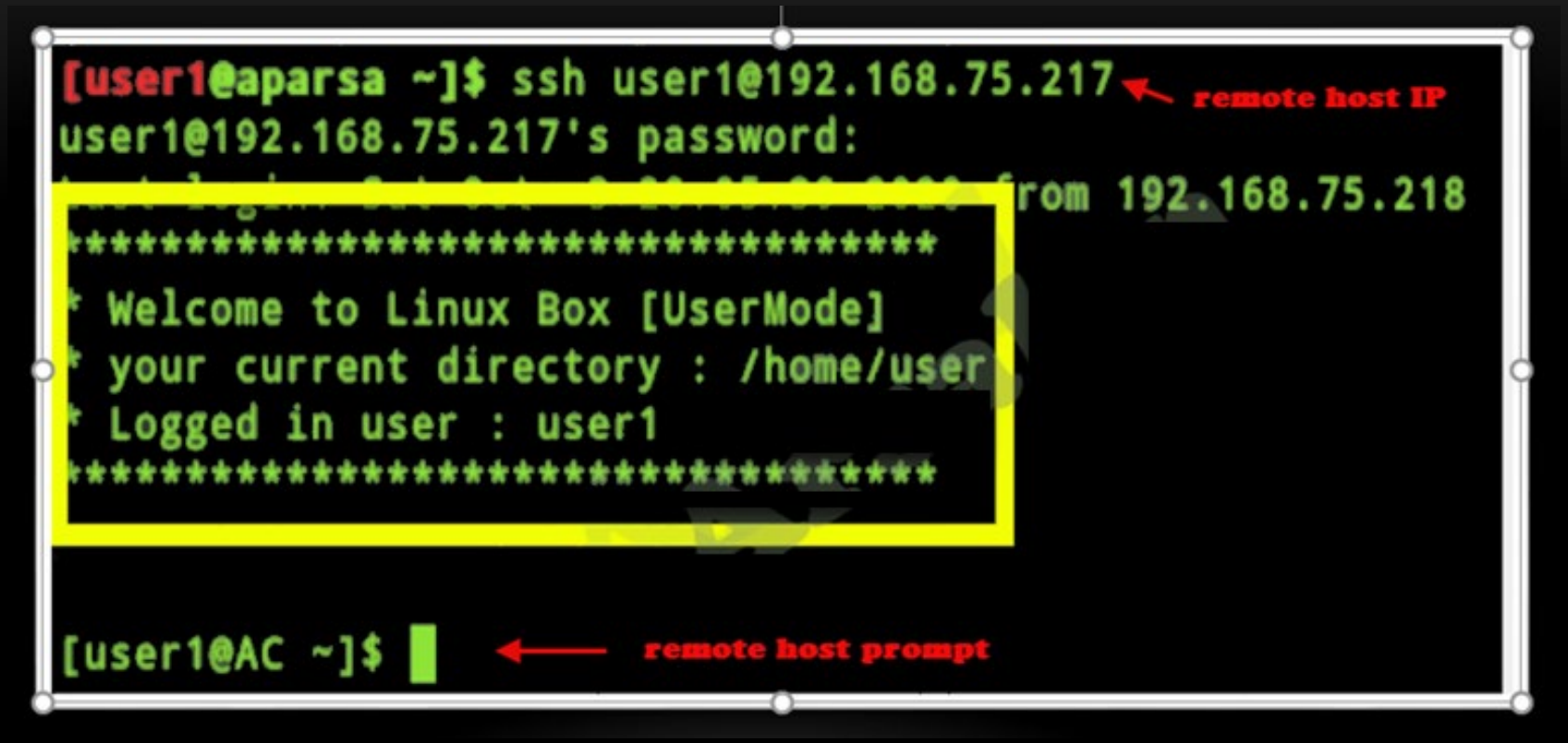
    1) $ service sshd start (or stop)
    2) $ systemctl start (or stop) sshd.service

- netstat (see what process is listening for instruction)

VERIFY “SSH DAEMON IS ACTIVE”



EXAMPLE OF HOW TO LOGIN REMOTELY TO ANOTHER HOST USING SSH



Note: the password is the user1 login password for remote host

SECURE COPY (SCP)

- **scp** uses ssh and rcp protocol for copying file securely. It is for Unix and Linux but you can find GUI version of secure shell too.
- The security level of scp is exactly as the level of security that ssh provides
- The difference is
  - SSH can execute command remotely, but SCP can't
  - With SSH you can control any machine remotely like routers, switches, smart devices, ICS, IoT, firewalls, and much more , but SCP just copy a file securely from one host to another
  - SCP uses SSH for its secure operation

HOW TO COPY FILE USING SCP

Above command, copy “dump.txt’ from current directory into remote host /home/user1/week6 directory.

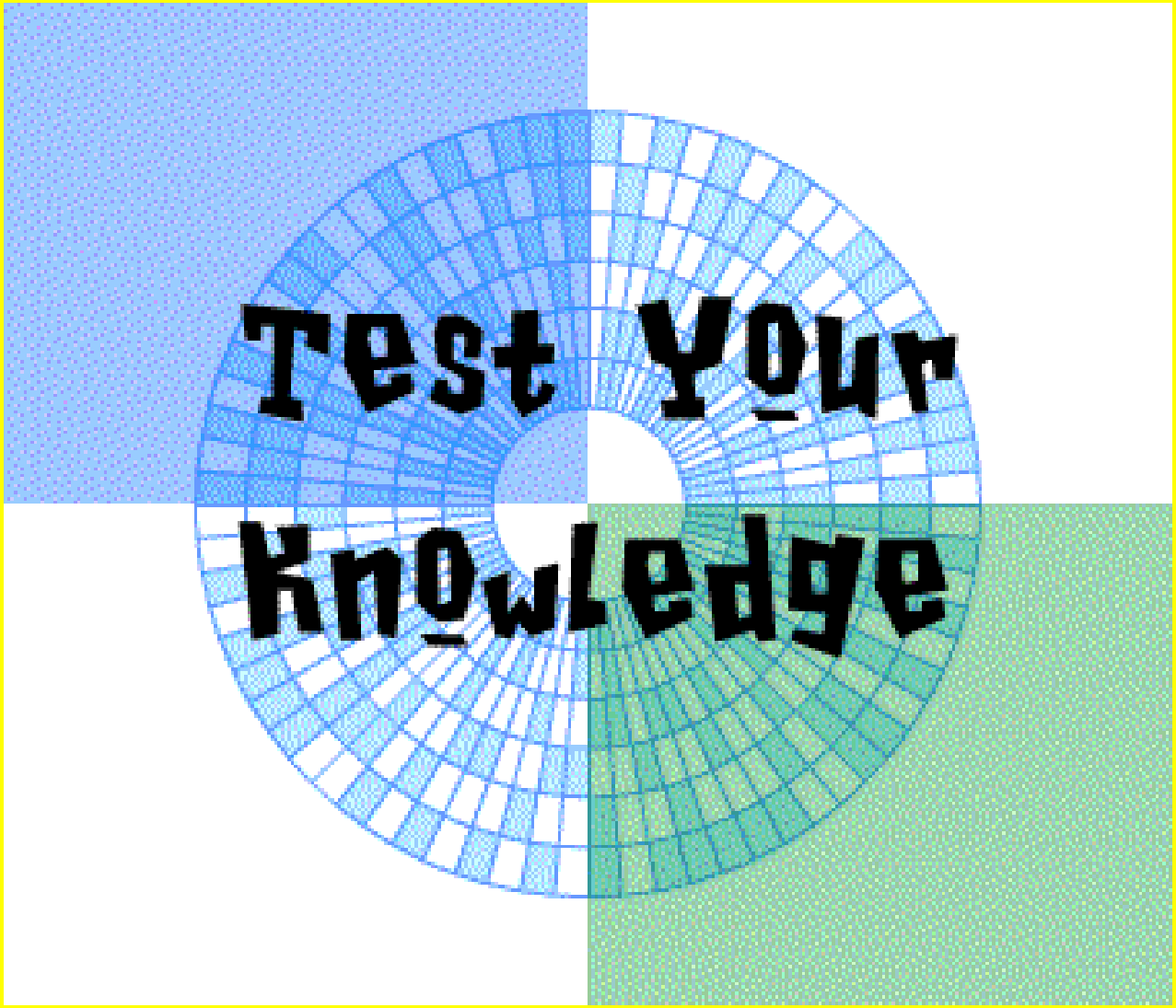
Note: don’t forget “ : “ right after the IP address of remote host

NOTE:

In your assignment 6, you will practice how to login remotely in to another host. Moreover, you will practice how to use SCP to copy different files into remote host. Make sure you read how to troubleshoot possible SSH & SCP challenge

In future Linux series courses, you will deploy SSH service for managing and administration of different enterprise assets

CHALLENGE



- Write a script such that it runs gedit process, then list the process and show the PID of gedit and then kill that process

WHAT TO DO FOR NEXT WEEK

- Complete Week 6 - Assignment 6
- Read the Week 6 contents
- Review the resources and practices