Mohammed Chowdhury

HW # 2

03/11/21

**A.Standard Directories and Files**

**1.Get a listing of your root directory**

-cd /bin

-ls

bin etc lib media root srv usr

boot home lib32 mnt run swapfile var

cdrom initrd.img lib64 opt sbin sys vmlinuz

dev initrd.img.old lost+found proc snap tmp vmlinuz.old

**2.Go to /bin directory.**

-cd /bin

**3.List its contents.**

-ls

bash fgrep ntfs-3g su

brltty findmnt ntfs-3g.probe sync

bunzip2 fuser ntfscat systemctl

busybox fusermount ntfscluster systemd

bzcat getfacl ntfscmp systemd-ask-password

bzcmp grep ntfsfallocate systemd-escape

**4.List 6 commands that you recognize.**

cp - makes copies of files and directories.

Mkdir - makes a new directory

Nano - opens the nano text editor

Ls - allows you to list all files and directories

Pwd - prints the working directory / shows the current directory we are in

Kill - allows you to send a signal to a process.

**5.Get a listing of the device directory. Do you recognize any device?**

-cd /dev

-ls

autofs input loop43 tty12 tty52 ttyS5

block kfd loop44 tty13 tty53 ttyS6

bsg kmsg loop45 tty14 tty54 ttyS7

btrfs-control kvm loop46 tty15 tty55 ttyS8

bus lightnvm loop47 tty16 tty56 ttyS9

cdrom log loop5 tty17 tty57 udmabuf

I personally do not recognize any device.

**6.Go to /etc directory.**

cd /etc

**7.Do a long listing; Mention a few files that you have already heard about.**

ls -l

drwxr-xr-x 3 root root 4096 Sep 7 2020 acpi

-rw-r--r-- 1 root root 3028 Feb 9 2019 adduser.conf

drwxr-xr-x 3 root root 4096 Sep 7 2020 alsa

drwxr-xr-x 2 root root 12288 Mar 11 21:20 alternatives

-rw-r--r-- 1 root root 401 May 29 2017 anacrontab

drwxr-xr-x 8 root root 4096 Sep 7 2020 apache2

...

A file that I recognize in the python3 directory and the kernel directory and the emacs directory.

**8.What is the most used permission? Whatdoes itmean?(read about permissions in Unix Handout)**

740, the read and write permissions are the most used.

**9.Using cat,check the passwd file or similar; look for yourself in the file**

The cat passwd command shows me all users on the system.

**B.Determine the absolute pathname for your home directory**

**10.Type:echo $HOME**

**11.Type:pwd**

Running this command gives me

/home/myUserName

**C.Shell(s) and Shell Environment variables**

**1.Check your default shell using: echo $SHELL**

/bin/bash

**2.Use the chsh command and find a list of available shells.**

The chsh command did not work so i tried this: sudo cat /etc/shells

# /etc/shells: valid login shells

/bin/sh

/bin/bash

/bin/rbash

/bin/dash

/bin/tcsh

/usr/bin/tcsh

**3.Change the current shell to a tcsh .**

Chsh

Enter the new value, or press ENTER for the defaultLogin Shell [/bin/bash]: usr/bin/tcsh

**4.Check your new shell. The change will not be listed until the next login**

I had to restart the PC to see change.

**5.Use the ps (process status –gives a lists of running processes). What do you observe?**

Before restarting PC:

PID TTY TIME CMD

5523 pts/0 00:00:00 bash

10929 pts/0 00:00:00 ps

After Restarting PC:

PID TTY TIME CMD

7071 pts/0 00:00:00 tcsh

7094 pts/0 00:00:00 ps

**D.Processes**

**1.Learn about the ps command using man (type man ps)**

**2.Give a list of possible states together with their significance. Identify your login shell.**

The possible states include sleeping, running, and zombies.

Sleeping → the process is waiting

Running → the process is currently running

Zombie → when the child process is never reaped and continuously running. The parent process must be terminated to kill a zombie process

The shell I am using is tcsh

**3.Type ps –l and explain the significance of :F, S, UID, PID, PPID, C, PRI, NI, ADDR, SZ, WCHAN, TTY, TIME, CMD fields.**

F → the f represents a flag. Ex. “4” represents the root user.

S-> the state processes. S = sleeping, R= Running, Z = zombie. T = traced

UID → this is the user ID.

PID → this is the process ID.

PPID → the is the parent process ID.

C → this is the CPU utilization.

PRI→ process priority (0 = highest , 127 = lowest )

NI→ nice value. (-20 = immediate priority , 19 = lowest priority)

ADDR → process memory address.

SZ→ size of process in memory.

WCHAN → what the process is waiting for while sleeping.

TTY → terminal the process is running on.

TIME → time process has taken on cpu.

CMD→ the command program or process.

**4.Use the top command to monitor the CPU activity in real time.It displays the status of the first 15 of the most CPU-intensive tasks on the system as well as the CPU activity. To stop the execution of top enter <ctrl-C>.**

**5.Give the total number of tasks, number of running processes, sleeping processes, stopped processes and zombies**.

There are about 17 processes.

Running → its is alternating between 1-2 processes.

Sleeping → about 16 processes.

Zombies → 0 processes.

**6.Identify the shell process. Use the “regular” kill command to terminate the shell.**

ps ux command was used to list all processes to get the PID of the CSH.

kill 7071 (pid of CSH was 7071)

Using a regular kill command did not do anything to the shell.

**7.Use the “sure kill” command to terminate the shell. Explain**

Kill -9 7071 successfully terminated the shell.

When using the kill command without given any flags, flag 15 (TERM) is used by default. However, it is not powerful enough to exit the shell. Hence the -9 (SIGKILL) flag was used to make sure the terminal was killed.

**Part E**

**1.Use Internet sources and give an overview of the command that is used in Windows for creating a process**

In windows, the create process API is used to create a new process.

**2.In a Unix environment, execute parent.c, child.c andorphan.c.**

Observe and understand the programs’ execution output. Extensively comment the output of the programs by relating the theory discussed in class, the meaning of the covered commands and the program listings.

Output for parent.c :

Process[16056]: Parent in execution ...

Process[16057]: child in execution ...

Process[16057]: child terminating ...

Process[16056]: Parent detects terminating child

Process[16056]: Parent terminating …

In the parent process what is happening is a new process was created using the fork() function.

If fork() returns 0 , it is a child process. If it is a parent process, it will return the PID of the parent. When the parent.c program is executed, the parent runs its process then tells the child to run its process. Using the wait() function, the parent checks to see if the child has been terminated or has been reaped,Once the child program terminates , the parent resumes its tasks.

Output for child.c:

Process[17721]: child in execution ...

Process[17721]: child terminating …

When the parent process is executed, the child process performs its tasks as the parent waits for it to finish. In this case, the child executes, sleeps/stays idle for 1 second then terminates.

Output for orphan.c :

I'm the original process with PID 17797 and PPID 15567.

I'm the parent process with PID 17797 and PPID 15567.

my child's PID 17798

PID 17797 terminates.

~/Desktop/Code/cs340Assignments/assignment2> I'm the child process with PID 17798 and PPID 1758.

PID 17798 terminates.

In this process, there is no wait function. Hence, the parent process does not wait for the child to terminate. This is why the child PPID does not match the parent PID, since it terminated before the child can finish.

Output for zombie.c:

i am the child with pid 19651

i am the parent with ppid 19650

I am the parent and my id is 19650



