Cs 340
Lecturer: Dr. Simina Fluture

Homework # 2 Unix

I WILL GRADE THIS HOMEWORK

Go over the homework requirements. In a separate file (use a text editor) cut and paste the commands that you typed (including the prompt) and the complete or partial output of these commands. The output of the commands should not contain more than six lines.

A. Standard Directories and Files

Directory: contains the names of files and/or sub-directories. Standard directories contain some special files.

Root Directory (/)
The root directory is the top of the file system. It is the master cabinet that contains all folders and files.

1. **Get a listing of your root directory.** (use, cd and ls –l)

/bin
The binary directory: contains executable files and most Unix commands.

2. **Go to /bin directory.** (use cd /bin)
3. **List its contents.**
4. **List 6 commands that you recognize.**

/dev
Device directory.

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

/etc
Contains commands and files for system administration. Usually a user is not allowed to change these files.

6. **Go to /etc directory.**
7. **Do a long listing; Mention a few files that you have already heard about.**
8. **What is the most used permission? What does it mean?**
9. **Using cat, check the passwd file or similar; look for yourself in the file.**

/lib
Contains a collection of related files for a given language in a single file called an **archive**.

/tmp
Contains temporary files.

/etc/passwd
Contains one line for every user on the system and describes that user.

Cs 340
Lecturer: Dr. Simina Fluture

B. Determine the absolute pathname for your home directory

10. Type:

```
echo $HOME
```

11. Type:

```
pwd
```

C. Shell(s) and Shell Environment variables

1. **Check your default shell using: echo $SHELL**
2. **Use the chsh command and find a list of available shells.**
3. **Change the current shell to a tcsh .**
4. **Check your new shell. The change will not be listed until the next login.**
5. **Use the ps (process status – gives a lists of running processes). What do you observe?**

Shell Environment variables

Bourne, Korn shell	C shell	
CDPATH	**cdpath**	alias names for directories accessed with cd
ENV		path along which Unix looks to find config. files
PS1	**prompt**	shell prompt that appears in the command line
PWD	**cwd**	name of current directory
HOME	**home**	the name of the user's home directory when the user logs
TERM		type of console terminal being used

D. Processes

Check the Unix Handout and go over the section about **Processes -section 17**.

The action of each shell, the mechanism of how it executes commands and programs, how it handles the command and program I/O and how it is programmed, are affected by the settings of certain environment variables.

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

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 task 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.**

6. **Identify the shell process. Use the "regular" kill command to terminate the shell.**
7. **Use the "sure kill" command to terminate the shell. Explain.**

PART E

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

2. In a Unix environment, execute parent.c, child.c and orphan.c as follows:
Note: upload first the 3 files in your venus home directory.

child and parent:
- compile the child and parent:

```
gcc parent.c –o parent
gcc child.c –o child
```

- run the parent in the current directory (the parent after the fork will call the child)
Don't worry about warning messages.

```
./parent
```

orphan:
- compile and run the orphan:

```
gcc orphan.c –o orphan
./orphan
```

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.

3. Write a very simple program that will show the possibility of having zombie processes.
Write a program named **zombie.c**

The main process will create a child.
The child prints something like: "I am the child with pid ….. and my parent has ppid …."
Next, the child will sleep for 1 second.
Child exits.

The parent will print: I am the parent and my id is… Next, the parent sleeps for 30 seconds.

Since the child ends first, and the parent didn't do wait( ), the child will be for a while in the zombie state. Run the parent in the background, so you can use the top command and identify the zombie, before the parent terminates.

Note: even if the parent terminates, the child is still a zombie. However the the init process reaps the zombies frequently.

You need to submit your homework solution on Blackboard *YourLastname*_H2
I will create a homework column named H2.
Your zip file should contain three files:
1. doc/txt file that cover parts A to E
2. zombie.c
3. snapshot of top command that shows the zombie process