BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE PILANI, K. K. BIRLA GOA CAMPUS

I SEMESTER 2012-2013

Operating Systems (CS C372 & IS C362)

Assignment 3

Due date 19/10/2012 (9.00 A.M) **Marks [5 + 15 = 20]**

Instructions:

(1) Please upload the assignment using **http://cc.bits-goa.ac.in/file**
(The file name should be <your id number>.tar.gz Example: **2012A7PS999G.tar.gz**)
You are allowed to upload only once.

(2) This is an **individual** assignment. Please see section 4b of handout for Malpractice regulations.

(3) The Assignment implementation should follow ADT with header files, implementation files, driver file & make file. No other implementation shall be considered for evaluation.

(4) The programming assignments will be graded according to the following criteria

- Completeness; does your program implement the whole assignment?
- Correctness; does your program provide the right output?
- Efficiency; have you chosen appropriate algorithms and data structures for the problem?
- Programming style (including documentation and program organization); is the program well designed and easy to understand?
- Viva.

DO NOT FORGET to include a README file (text only) in your tar.gz file with the following contents.

General README instructions

In the directory you turn in (please upload the assignment as a tar.gz file), you must have a text-only file called README, in which you will cover AT LEAST the following:

1. Your name. If you interacted significantly with others indicate this as well.
2. A list of all files in the directory and a short description of each.
3. HOW TO COMPILE your program and HOW TO USE (execute) your program.
4. A description of the structure of your program.
5. In case you have not completed the assignment, you should mention in significant detail:
   - What you have and have not done
   - Why you did not manage to complete your assignment (greatest difficulties)
     This will allow us to give you partial credit for the things you have completed.
6. Document any bugs of your program that you know of. Run-time errors will cost you fewer points if you document them and you show that you know their cause. Also describe what you would have done to correct them, if you had more time to work on your project.

Please refer section 4b of the handout to know more about Malpractice regulations of the course.

QUESTION 1: *Experiencing exec() Commands in Linux!!!*

Write a program for the following.
Input File Format:
<No. of commands = N>
<Command_1> <dependency on> <environment path variable> <options>
<Command_2> <dependency on> <environment path variable> <options>
<Command_3> <dependency on> <environment path variable> <options>
.
.
.
.
<Command_N> <dependency on> <environment path variable> <options>

If the environment path variable is "**–**" then you need to set the path as **"/bin: /sbin/: /etc/: /home/usr/**"

Assumptions:

- You are expected to have super user privileges at all times.
- Each command will depend on not more than one command.

Procedure:

The main process should read the input file using filename as the argument to the main function. While reading the input file, the main process should **create two 2-D character arrays arrA and arrB** for storing the commands in it. The grouping should be done as mentioned in the steps below.

(A) A dependency indicates whether a command depends on any other command and must execute only after that command has finished execution.
For example: If the entry says
ls ps /bin/ –l
This indicates that ls should execute only after ps has been executed.

(B) If a **command depends on another command**, it should be added to the array containing the dependency command entry.
So, for the above example, ls should be added to the array that contains ps and the entry should be anywhere after the entry of ps.

(C) If a **command is not dependent on any other command**, then it must be inserted into the smaller array among the two.

(D) After grouping all the commands according to Step A to Step C, you will get two 2-D arrays containing commands to be executed after satisfying all their dependencies.

(E) Now, the main process should fork two child processes.

(F) Child 1 will execute commands from arrA sequentially and Child 2 will execute commands from arrB sequentially. Child 1 and Child 2 will execute concurrently.

(G) For each command, the child processes will fork another process for the execution which will use the execv() system call to execute the command along with the environment path variable specified in the input file.

(H) Make sure you are not creating orphan process(es) while executing. i.e. all the child processes should finish before its parent terminates normally.

QUESTION 2: *Wannabe Hacker !!!!*

Since you are going to pursue a major in computers, you must be aware of how hacking is done (for your own benefit). We present to you a very sneaky way into "Some one's" user space where in you can edit your own marks for your assignment.

Aim: Change your Test 1 marks in "Some one's" user space at your own will without any penalty.

Input Format:

The input must be taken as follows

<filename> <ID No.> <New marks>

This input must be taken as arguments to the main function.

Format of the Marks file:

<ID No.> <\t> <Marks>

Assumptions:

- You must have super user privileges at all times.
- **Create two groups** 1.) Evaluator, 2.) Student *(please refer to figure1 at the end for the exact hierarchy expected of the directories)*
- There will be **two evaluators** in the evaluator group.
- Individual evaluator will have its own directories and subdirectories. One of them will contain your marks file.
- You should write **your code in the Student's directory** and not in the Evaluator's.
- There **may be a directory** of the same name as that of the file you want to search.
- The code will **start searching** the marks file **from the Evaluator directory**, down the hierarchy.
- There will be **no space** in the name of any file or directory.

- You **can use 'chdir'** system call as **'cd' command does not work** in exec(). You must use exec() series system calls for executing the rest of the commands like 'pwd', 'ls', 'grep', 'rm'.

Procedure to change:

- The code should execute the following command
  ls <options> | grep <options> expression
  You are expected to fork two processes (one for ls and the second one for grep taking the input as output of ls command) that will execute the above command.

- After execution of this command, check whether you have found the given marks file.

- If you haven't found the file in the current working directory (cwd), the process should access the shared space to check whether the flag is set. The flag will be set only if some other process already found the file. So this process can exit without proceeding further. If the flag is not set, fork as many number of children as the number of directories in the cwd.

- It is possible that some of the directories in your cwd have read only permissions. If this is the case, you are expected to change these permissions to read –write – execute mode for group, user and others.
  This can be done using the chmod() system call. After changing the permissions, you will be allowed to traverse down the hierarchy.

- In each child process, change the cwd to the directory for which the child has been forked in the previous step.

- Now, each of the child processes must search for the file in their respective cwd simultaneously.

- At any point, if the file is found,
  - The process will access the shared memory to check if any other process has already found the file and there is a flag set to 1 in the shared space. If the flag is already set, the process should readily terminate without continuing further.
  - If the flag is not set in the shared space, the process will,
    - Execute pwd command to get the path.
    - Write this path in a shared memory.
    - Set the flag to 1.

- A process will terminate normally if it does not find any directory after the execution of the command stated in the first step.

- Till all the children finish their execution, the main process must wait (No orphans should exist in the system).

- After the termination of all child processes, the main process will access the shared memory to change your marks in the file whose path is present.

- First search your ID no. in the given file and then edit your marks corresponding to it. You may need the creation of temporary file while the editing is being done.

- After your marks have been changed in the file, your main process should terminate.

- The following figure gives the sample directory hierarchy for your reference.

- The marks file can be in any of the evaluator's directory.

- The figure below shows just a sample example for your understanding and not any concrete directory structure. So, please do not hardcode with reference to the figure below. Your code is expected to be generic.

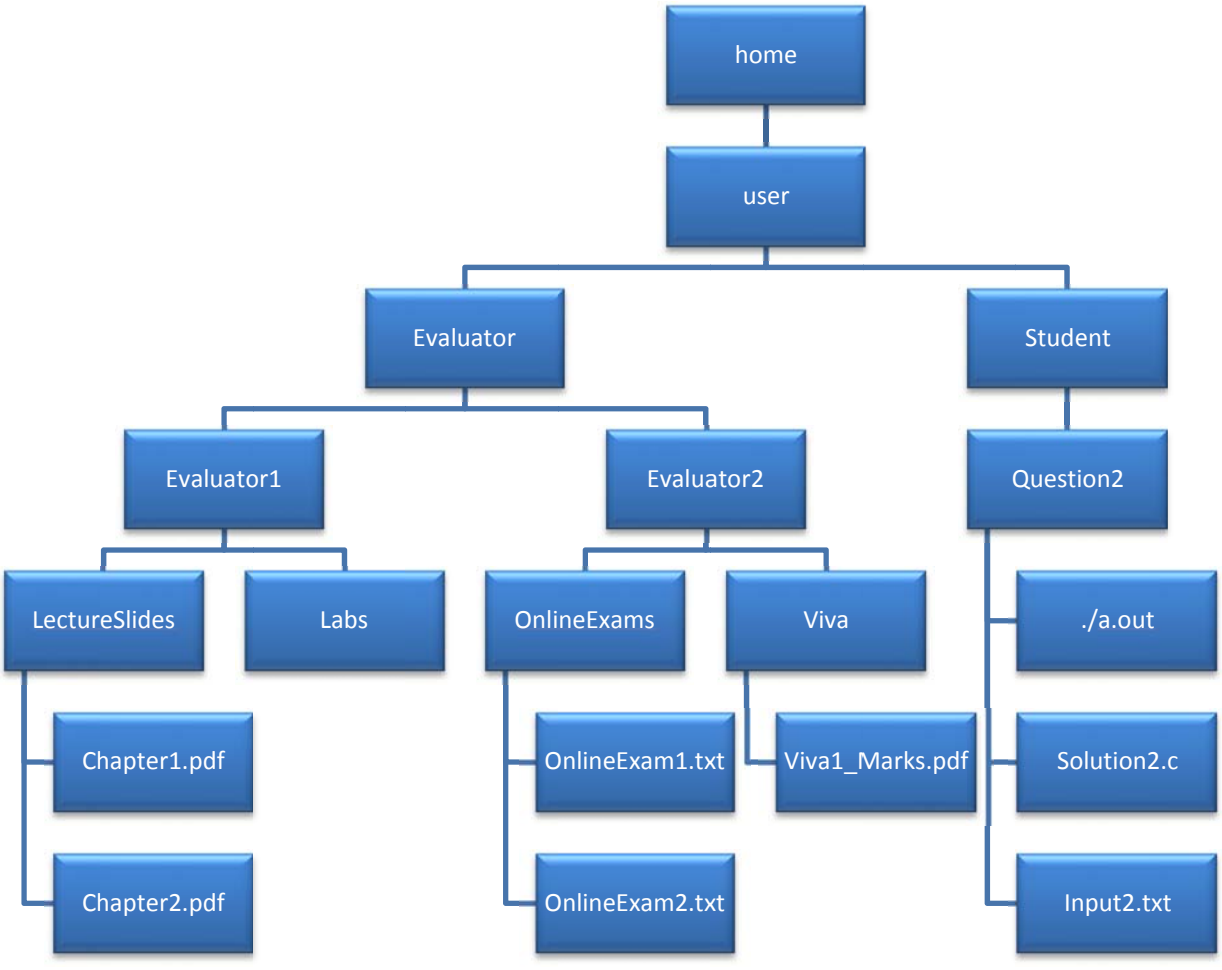

Figure 1 : Sample directory hierarchy with groups evaluator and student

QUESTION

You have to do the above procedure twice

(A) First using dup and dup2 (B) Second using Pipe / Tee

Your code must also delete all the temporary files that have been created in the above procedure.

P.S.:

- Please be realistic while giving yourself marks for Test 1 so that “no user” will get suspicious. If you have taken help from anyone, please be fair enough.

- Please remove all unnecessary printfs, so that “no user” will get suspicious and you will end up safely with your marks.