Operating Systems

Grado en Informática 2023/2024

Lab Assignment 3: Processes

We continue to code the shell we started in the first lab assignment. We'll add the following commands. Check the supplied shell to check the workings and exact sintax for the commands. (You can use the help command, "command -?" or "command -help" to get help):

uid	views (or changes) the process's credentials
showvar	shows the value of an environment variable
changevar	changes the value of an environment variable
subsvar	changes one environmet variable for other
showenv	shows the process environment
fork	the shell does the fork system call and waits for its child to end
exec	executes a program (with arguments) without creating a new process. It is not neccesary to do the priority (@pri) nor the reduced environment (VAR1 VAR2..) parts
jobs	lists background processes
deljobs	deletes background processes from the list
job	shows info on a background process. Can change it to foreground
*****	For anything that is not a shell command, the shell will asume it is an external program (in the **format for execution** described below) and will create a process to execute it
***** &	For anything that is not a shell command, the shell will asume it is an external program (in the **format for execution** described below) and will create a process IN THE BACKGROUND to execute it. The process will be added to the list of background processes. (the & indicates execition in background, must not be passed to the program as a argument).

IMPORTANT:

We have to implement (same list type as we used before) a list of processes executing in the background. For each process we must show

- Its PID
- Date and time of launching
- Status (FINISHED, STOPPED, SIGNALED or ACTIVE) (with return value/signal when relevant)
- Its command line
- Its priority

The shell commands ***jobs*** and ***deljobs***, show and manipulate that list. Only processes launched from the shell to execute in the background will be added to the list.

We insert processes in the list as ACTIVE. We should update, with the waitpid system call, the status of processes before printing. Note that the waitpid system call reports status changes, not the state itself, in fact, it only reports once a process finished.

```
pid_t waitpid (pid_t pid, int * wstatus, int options);
```

The parameter *wstatus* ONLY has a meaninful value in the case waitpid returs the pid.

Priority (as it can change) should be obtained at the time of printing so it is not necessary to store it it the list.

Execution in foreground means the parent process **waits** for the child to finish before continuing.

Execution in the background means the parent continues to execute concurrently with the child: i**t does not wait** for its child to finsh.

******* format for execution**

The format for creating process that executes a program is

executable arg1 arg2.....[&]

items inside brackes [] are optional

- **executable** is the name of the executable file to be executed
- **arg1 arg2 ...** are the arguments passed to the executable. (number of them is undefined)
- **&** if present means that execution is to be in the background
- This format, with the execption of the &, is the same to be used with the ***exec*** command to execute without creating process

Examples

-> ls

executes, creating a process in the foreground, the program ls

-> ls -l -a /home

executes, creating a process in the foreground, 'ls -l -a /home '

-> xterm -fg yellow -e /bin/bash

executes, creating a process in the foreground, 'xterm -fg yellow -e /bin/bash'

-> **xterm -fg green -bg black -e /usr/local/bin/ksh &**

executes creating a process in the background 'xterm -fg green -bg black -e /usr/local/bin/ksh'

-> exec xterm -fg white

executes without creating a new process 'xterm -fg white'

We'll use the *setuid* systemt call to change the user credential. Under normal circunstances the real and effective credentials will be the same, so no changes in the proccess credentials are allowed. To check this part of the lab assigment we'll have to

- Give the executable the setuid execution bit rwsr-xr-x (4755)

- Have it executed by another user so that the real credential is that of the user executing the file and the saved and efective credentials are those of the owner of the file. (It's a good idea to place the executable file on a writable directory for both of those users: `/tmp`.)
- The *setuid* system calls behave direrently for the `root` user, son we must not use that account to check this part of the lab assigment)

REMEMBER:

- Information on the system calls and library functions needed to code these programs is available through man: fork, exec, waitpid, getenv, putenv, getpriority,....
- A reference shell is provided (for various platforms) for students to check how the shell should perform. This program should be checked to find out the syntax and behaviour of the various commands. **PLEASE DOWNLOAD THE LATEST VERSION**
- The program should compile cleanly (produce no warnings even when compiling with **gcc -Wall)**
- These programs can have no memory leaks (you can use valgrind to check)
- When the program cannot perform its task (for whatever reason, for example, lack of privileges) it should inform the user
- All input and output is done through the standard input and output
- Errors should be trated as in the previous lab assignments
- An additional C file is provided with some useful functions

WORK SUBMISSION

- Work must be done in pairs.
- Moodle will be used to submit the source code: a zipfile containing a directory named P3 where all the source files of the lab assignment reside
- The name of the main program will be p3.c, Program must be able to be compiled with gcc p3.c, Optionally a Makefile can be supplied so that all of the source code can be compiled with just make. Should that be the case, the compiled program should be called p3
- ONLY ONE OF THE MEMBERS OF THE GROUP will submit the source code. The names and logins of all the members of the group should appear in the source code of the main programs (at the top of the file)
- Works submited not conforming to these rules will be disregarded.
- DEADLINE: 23:00, **THURSDAY December the 14th, 2023**