Programmer's View - Creating Processes

Christian Khoury

1 Creating and Running a Process (1) - fork

1. Read the *fork, getpid, and getppid* manuals.
2. What happens after a *fork* call ? How are parent and child differentiated ?
3. Write a small C program in which the parent process creates a child process and each displays a different message : *I'm the parent* vs *I'm the child*. Display the process id and the parent process id for every running process.
4. Is data shared between parent and child ?

```
int i = 5;

if (fork() == 0)
  { // I'm the
  ...
  i++;
} else {
   // I'm the ...
   sleep(3); // sleep for 3 seconds ; why ?
   printf("%d\n", i); }
```

   Why does this snippet helps in answering the previous question ?

5. Is it possible to create more than one child process ? Show how using a simple program that creates 2 children for the 1st-level process (main parent) and a child for one of the 2nd-level processes (children).

2 Creating and Running a Process (2) - exec

When we create a child process, we usually want to run a different application, and that can be done using the *exec* family of functions !

1. *man 3 exec*

2. use any of these functions to run “firefox” or any other application of your choice; Is the process id of the new running application different from the original one ? Explain how you figured this out.

```
int main(){
  // display the process id
  // simply use any exec call !
}
```

3. Is data shared by the parent and child processes and to what extent ? Explain. Use for that matter :
    a. The documentation available (does it mention anything about data sharing ?)
    b. Your user experience in operating systems ; do you usually share data between your running apps (word, game, browser, …) ?
    c. Write some code (if possible) to prove it.

4. Explain what happens in the following program. What is the main difference with the previous version (question 2) ?

```
int i = 5;

if (fork()== 0){

  // write an exec call

  i++;

  printf("%d\n", i); // how is this line handled ?

} else {
  // display the process id of this process
}
```

3 Final piece of the puzzle – A simple shell

1. Write a simple program that prompts the user for a command (without arguments, i.e.,ls, ps, firefox, …) to run and executes it,

```
awaits the termination of the current
execution (investigate the wait function)
then prompts for a new command indefinitely.
```