# 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 processand 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
 printf("%d\n", i);

}
```

```
 Why do we logically need the « sleep(3) »
in this code ?
```

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 yourchoice; 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.
4. Explain what happens in the following program. What is the main difference with the previous version ?

```
int main()
{
   int i = 5;
   if (fork() == 0) {
      // write an exec call here
      printf("%c\n", i); // is this line executed ? why ?
   } else
      // display the process id here
}
```

## 3 Writing your own shell

1. Read the *system* function documentation (https://man7.org/linux/man-pages/man3/system.3.html)

2. Implement your own system function (call it « mySystem »)

3. Write a program that displays the following menu in a loop :
   1- run a program
   2- kill a process (*hint : lookup the kill manual)*
   3- list the files in the current folder (*hint : lookup the ls manual*)
   4- quit

   Use the «mySystem » function to implement the different options of your menu (except for quit of course).