Programmer's View (1) - Creating Processes

Christian Khoury

In this lab, we will discover the programming facet of the Linux operating system using the C/C++ language. Throughout this lab, test the different commands/expressions and write down your comments and what ever you've learnt (be concise) in a pdf or word-like document then put it on campus (use your lastnames to label the file). Zip the source files and the report using *tar*.

1 Compiling under Linux

Everything is explained in the file GCC and GDB on campus.ece.fr

Using your usual text editor (vi, emacs, gedit, ...), write a simple "hello world" program.

```
#include <stdio.h>

int main() {
  printf("Hello World!");
}
```

1. *gcc -o execName program.c*
2. *./execName*
3. *gcc -g -o execName program.c* Why do we use the "-g" option ?
4. *gdb execName* and then use the following simple commands
   - *list*
   - choose a line and toggle the breakpoint using *break lineNumber*
   - then run the process using *run arglist*
   - Is there a help command ? other types of commands ?

2 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
  printf("%d\n", i); // what happens here ?? Explain
}
```

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).

3 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.

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

```
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
}
```