# Evaluating Processes with Fork and Wait

## Introduction

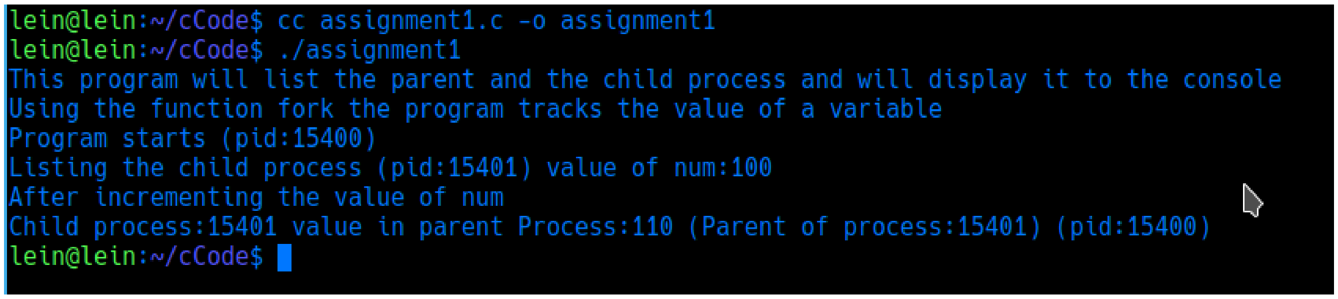
In the process of virtualization and the creation of the illusion by the OS that there are multiple virtual machines out of only one single hardware, modern operating systems implement data structures in order to keep track of the execution and an identification is assigned for each one of the processes known as PID which we will briefly review by using the built-in functions fork and wait.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>

int main(int argc, char const *argv[])
{
	//List parent and child process
	printf("This program will list the parent and the child process and will display it to the console\
n");
	printf("Using the function fork the program tracks the value of a variable\n");
	printf("Program starts (pid:%d)\n", (int) getpid() );
	int num = 100;
	int rc = fork();
	if(rc < 0){
		fprintf(stderr, "fork failed\n");
		exit(1);
	} else if( rc == 0 ) {
		printf("Listing the child process (pid:%d) value of num:%d \n", (int) getpid(), num );
	} else {
		int rc_wait = wait(NULL);
		num += 10;
		printf("After incrementing the value of num\n");
		printf("Child process:%d value in parent Process:%d (Parent of process:%d) (pid:%d)\
n",

		rc, num, rc_wait, (int) getpid() );
	}
	return 0;
}
```

The program implemented above gets the pid of the initial process when the value of the variable used is assigned to 100 in, in the output shown below, we can see the pid is 15401 when the value is 100. After the main program is running the value of the variable is incremented, the newly created process receives only a reference to the previous process when the fork() is called.
We can easily perceive that some common things, The child process always is displayed first, then the fork references the child process and a new  pid is assigned in our example the pid is 15401 however, when the child process is completed we evaluated the parent process which remains the same 15400 even though it holds a reference to the variable after it has been incremented.



```
lein@lein:~/cCode$ cc assignment1.c -o assignment1
lein@lein:~/cCode$ ./assignment1
This program will list the parent and the child process and will display it to the console
Using the function fork the program tracks the value of a variable
Program starts (pid:15400)
Listing the child process (pid:15401) value of num:100
After incrementing the value of num
Child process:15401 value in parent Process:110 (Parent of process:15401) (pid:15400)
lein@lein:~/cCode$
```

Conclusions
My thoughts on the assignment are most concerning to the programming language we are using through the course, I have found it to be a bit challenging which has forced me to invest more time getting the hang of so I will be better able to understand the code examples and snippets used as a reference for the course. On the other hand, I can already see how the course is helping me to get better at using my OS which is a Linux distribution and most of the code I am able to run it without the need to install additional dependencies or third-party libraries I just need to open a shell session on my laptop and easily I have been able to compile and execute the code. The assignment helped me to identify some of the most used functions in my OS in order to determine what my hardware is doing at any given time.
Reference list:

Arpaci-Dusseau, R. & Arpaci-Dusseau, A. (2012). *Operating Systems: Three Easy Pieces. Madison, WI: University of Wisconsin-Madison*. Available at http://pages.cs.wisc.edu/~remzi/OSTEP/