Assignment 1 : Operation on Process

**Objective :**

• This assignment is intended to learn how to create, work with and manipulate processes in Linux. You are expected to refer to the text book and references mentioned in the course website before you start the lab.

**Assignments:**

1. Write a ***CPU bound* C** program and a ***I/O bound* C** program (e.g. use a number of *printf* statements within a *while(1)* loop). Compile and execute both of them. Observe the effect of their CPU share using the *top* display and comment.
2. Write **a program in C** that creates a child process, waits for the termination of the child and lists its PID.
3. Compile and run the program code for***asgn1.c*** and record your observations. Perform the modification mentioned and answer the questions that follow.

(a) Comment the inner loop in both the if and the else blocks, compile and run program code for *asgn1.c* again. Record your observations.

(b) Do you find any difference in the output. If not, then what do you think is the role of the inner loop in both if and the else blocks ?

(c) Modify code for asgn1.c in order to make the child process finish before the parent process starts

1. Create a file named my\_file.txt that contains the following four lines :

*Child 1 reads this line*

*Child 2 reads this line*

*Child 3 reads this line*

*Child 4 reads this line*

Write **a C program that forks four other processes**. After forking the parent process goes into wait state and waits for the children to finish their execution. Each child process reads a line from the file my file.txt (Child 1 reads line 1, child 2 reads line 2, child 3 reads line 3 and child 4 reads line 4 ) and each prints the respective line. The lines can be printed in any order.

1. Write two programs **file1.c** and **file2.c**. Program **file1.c uses these** :

(a) **fork()** to launch another process

(b) **exec()** to replace the program driving this process, while supplying arguments to file2.c to complete its execution

(c) **wait()** to complete the execution of the child process

(d) file1.c takes two arguments x (a number less than 1) and n (number of terms to be added, 1 or more). For example: file1 0.5 5

(e) When the child proces finishes, the parent prints:

**Parent(PID=yyy) : Done**

Program **file2.c requires** two arguments to obtain the approximate value of ex by adding the first n terms in the relation : ex = 1+x+x2/2!+x3/3!+....... and prints the result in the format:

**Child(PID=yyy) : For x = 0.5 the first 5 terms yields 1.6484375**

*Hint : Child-specific processing immediately following the fork() command should load file2.c into the newly created process using the exec() command. This exec() command should also pass 2 arguments to the child. Refer to the man page of exec() command to know how to pass on arguments to the child process. Parent-specific processing should ensure that the parent will wait() for the child- specific processing to complete.*

*作业1：流程操作*

*客观的*

*•本作业旨在学习如何在Linux中创建、使用和操作进程。在开始实验室之前，你应该参考课程网站上提到的课本和参考资料。*

*作业：*

*1.编写一个CPU绑定的C程序和一个I/O绑定的C程（例如，在while（1）循环中使用多个printf语句）。编译并执行这两者。使用顶部显示和注释来观察它们的CPU共享的效果。*

*2.用C语言编写一个程序，创建一个子进程，等待子进程的终止并列出其PID。*

*3.编译并运行asgn1.c的程序代码，并记录您的观察结果。执行上述修改并回答以下问题。*

*（a） 对if和else块中的内部循环进行注释，编译并再次运行asgn1.c的程序代码。记录您的观察结果。*

*（b） 你发现输出有什么不同吗。如果不是，那么你认为内环在If和else块中的作用是什么？*

*（c） 修改asgn1.c的代码，使子进程在父进程启动之前完成*

*4.创建一个名为my\_file.txt的文件，该文件包含以下四行：*

*孩子1读这行*

*孩子2读这行*

*孩子3读这行*

*孩子4读这行*

*编写一个C程序来派生其他四个进程。分叉后，父进程进入等待状态，等待子进程完成执行。每个子进程从文件my file.txt中读取一行（子进程1读取第1行，子进程2读取第2行，子过程3读取第3行，子程序4读取第4行），并分别打印相应的行。行可以按任何顺序打印。*

*5.编写两个程序file1.c和file2.c。file1.c程序使用这些程序：*

*（a） fork（）启动另一个进程*

*（b） exec（）替换驱动该进程的程序，同时向file2.c提供参数以完成其执行*

*（c） wait（）完成子进程的执行*

*（d） file1.c接受两个参数x（小于1的数字）和n（要添加的项数，1或更多）。例如：file1 0.5 5*

*（e） 当子进程完成时，父进程将打印：*

*父级（PID=yyy）：完成*