LAB6_2019136067_배수빈

LAB6_1 : 컴파일 프로그램 만들기

```
xubin@DESKTOP-G3939FB:~/SystemProgramming/Homework/Lab6/Lab6_1$ ls
Lab6_1.c  Makefile  child.c  tmp.txt
xubin@DESKTOP-G3939FB:~/SystemProgramming/Homework/Lab6/Lab6_1$ make
gcc           -g -Wall    -c -o Lab6_1.o Lab6_1.c
gcc           -o Lab6_1.out Lab6_1.o  # Lab6_1.out : (automatic variables) current target name
xubin@DESKTOP-G3939FB:~/SystemProgramming/Homework/Lab6/Lab6_1$ ./Lab6_1.out child.c child.out
xubin@DESKTOP-G3939FB:~/SystemProgramming/Homework/Lab6/Lab6_1$ ls
Lab6_1.c  Lab6_1.o  Lab6_1.out  Makefile  child.c  child.out  tmp.txt
xubin@DESKTOP-G3939FB:~/SystemProgramming/Homework/Lab6/Lab6_1$ ./child.out tmp.txt res.txt
xubin@DESKTOP-G3939FB:~/SystemProgramming/Homework/Lab6/Lab6_1$ cat res.txt
HELLO
```

설명 및 소감

execlp의 인자에서 헤맸던 문제이다. man-page형태로 exec함수들에 대해 설명해준 내용들이 내가 이해한 내용들과 조금 달랐다. 처음 인자로 들어가는 file이 내가 실행하고자하는 파일을 말하는줄 알았는데 ls, gcc와 같은것들을 말하는것 같았다. 인자에서 헤맸던것 빼면 평소 컴파일하는 순서대로 인자값으로 넣으면 잘 풀리는 문제였다.

LAB6_2 : 부모님이 누구니

```
xubin@DESKTOP-G3939FB:~/SystemProgramming/Homework/Lab6/Lab6_2$ ./Lab6_2.out
Second Child Process
My PID: 201
My Parent's PID: 199

First Child Process
My PID: 200
My Parent's PID: 199

Parent Process
My Child's PID: 200, 201
My PID: 199
My Parent's PID: 9
```

설명 및 소감

이번문제를 풀면서 fork()를 하면 fork된 자식과 부모는 '각각 서로 독립적으로 프로그램을 실행한다'하는 개념을 확실히 이해할 수 있었다. 처음에 자식의 pid값들을 배열로 저장을 했었는데 부모에서 출력하면 계속 초기값이 출력되었었다. 왜 그런지 생각해보며 fork면 각각 독립적으로 실행되기 때문에 자식에서 저장한 배열 값들이 전혀 소용없다는것을 깨닫고 문제를 풀 수있었다.

LAB6_3 : myTimer 만들기

```
xubin@DESKTOP-G3939FB:~/SystemProgramming/Homework/Lab6/Lab6_3$ ./Lab6_3.out 1 1000
[Serial start] Thu Nov  4 23:26:29 2021
[Serial] found 168 primes
[Serial end] Thu Nov  4 23:26:29 2021
[Paralle start] Thu Nov  4 23:26:29 2021
[pid = 0] I found 95 prime numbers between (1 ~ 500)
[pid = 0] takes 0.154 ms
[Proc.0 end] Thu Nov  4 23:26:29 2021
[pid = 642] I found 73 prime numbers between (501 ~ 1000)
[pid = 642] takes 0.33 ms
[Proc.642 end] Thu Nov  4 23:26:29 2021
```

Single Algorithm과 Parallel Algorithm의 성능비교

확실히 Parallel Algorithm이 일을 나눠서 두명이 처리해서 성능이 좋다. 함수처럼 같은 코드를 이용하는데 그것을 동시에 진행할 수 있어 효율이 확실히 좋아진다는 사실이 너무 좋은 것 같다.

Parallel Algorithm에 대한 고찰

이런 문제를 짤때 항상 코드적인 부분에서 시간을 줄이려고만 생각했는데 fork를 통해 프로세서 관점에서 더 넓혀서 생각하니 프로세스를 이용해서 시간을 줄일 수 도 있어 놀라웠다.

설명 및 소감

pdf의 예시가 잘못나와있어서 코드가 잘못된 줄 알고 고생을 좀 했다. 배운내용을 활용하는 측면에 있어서는 특별히 어려운 부분은 없었던것 같다. 그리고 프로세스를 다루는 것이 아직까지는 완벽히 이해를 못한것 같아서 계속 사용하면서 익숙해지고 자유자재로 쓸 수 있도록 노력해야겠다.