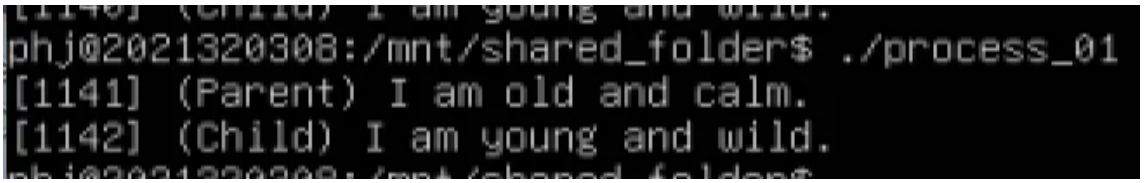
OS 실습 과제 3

process_01

</1>	default
</2>	case 0
</3>	getpid()
</4>	0

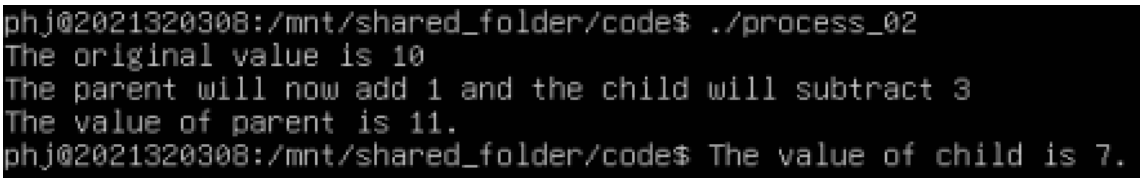
결과 스크린샷:



process_02

</1>	pid > 0
</2>	pid == 0
</3>	0
</4>	val

결과 스크린샷:



process_03

</1>	“/bin/pwd”, “pwd”, NULL

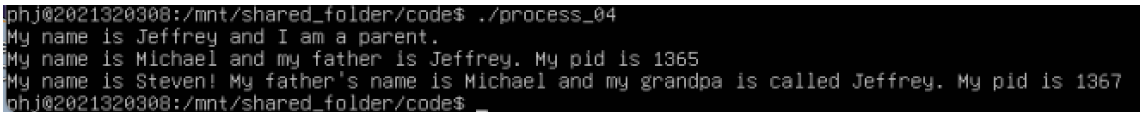
결과 스크린샷:

process_04

</1>	default
</2>	case 0
</3>	pid2
</4>	child_name, name, getpid()

</5>	case 0
</6>	grandchild_name, child_name, name, getpid()

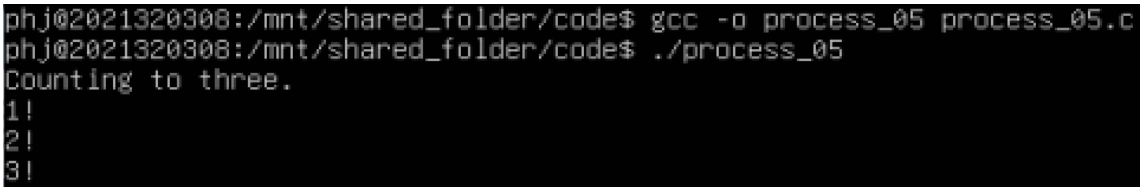
결과 스크린샷:



process_05

</1>	pid > 0
</2>	&status
</3>	status/256
</4>	pid == 0
</5>	3

결과 스크린샷:

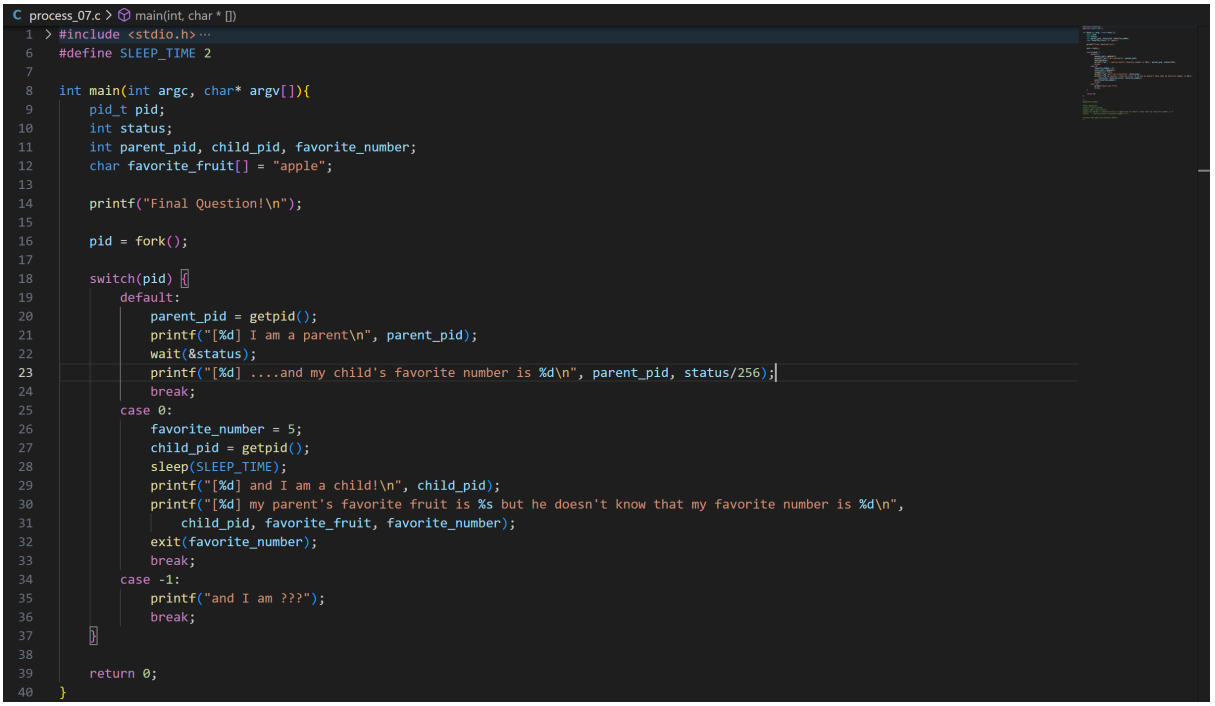


process_06

</1>	pid > 0
</2>	pid2 > 0
</3>	pid, &status, 0
</4>	pid2, &status, 0
</5>	pid2 == 0
</6>	pid == 0

결과 스크린샷:

process_07



```c
process_07.c > main(int, char * [])
#include <stdio.h> ···
#define SLEEP_TIME 2

int main(int argc, char* argv[]){
    pid_t pid;
    int status;
    int parent_pid, child_pid, favorite_number;
    char favorite_fruit[] = "apple";

    printf("Final Question!\n");

    pid = fork();

    switch(pid) {
        default:
            parent_pid = getpid();
            printf("[%d] I am a parent\n", parent_pid);
            wait(&status);
            printf("[%d] ....and my child's favorite number is %d\n", parent_pid, status/256);
            break;
        case 0:
            favorite_number = 5;
            child_pid = getpid();
            sleep(SLEEP_TIME);
            printf("[%d] and I am a child!\n", child_pid);
            printf("[%d] my parent's favorite fruit is %s but he doesn't know that my favorite number is %d\n",
                child_pid, favorite_fruit, favorite_number);
            exit(favorite_number);
            break;
        case -1:
            printf("and I am ???");
            break;
    }

    return 0;
}
```

설명:

- **line 1~12**: 이 main함수에 필요한 헤더파일들과 매크로 “SLEEP_TIME”, main함수에서 사용하는 지역 변수들.
- **line 16**: fork()를 호출하여 pid에 리턴값을 넣는다. 이는 parent 프로세스의 호출이다.
- **line 18**: pid의 값에 따라 스위치 문을 실행한다.
- **line 19~21**: default는 {pid > 0}인 경우로써, parent의 scope에 해당한다. 따라서 parent_pid = getpid() 에는 의도한대로 parent의 pid가 저장된다.
- **line 22**: child 프로세스가 종료되길 기다린다. 이때, child가 exit()로 리턴해주는 값을 status에 저장한다.
- **line 23**: child가 exit()로 리턴하는 값을 parent가 받게되면 상위 8 bit에 저장하기 위해 256이 곱해진다. 따라서 status/256 을 사용한다.
- **line 25~27**: child는 fork()로부터 0을 return 받는다. 즉, case 0는 child의 scope를 나타낸다. 따라서 child_pid = getpid() 에는 의도한대로 child의 pid가 저장된다.
- **line 28**: parent의 printf 문(”I am a parent”)이 먼저 안전하게 출력될 수 있도록 child에서 sleep()을 호출한다.
- **line 29~31**: child의 printf 문을 출력한다.
- **line 32**: exit()로 favorite_number를 넘겨준다. 이를 통해 parent 프로세스는 child의 favorite_number를 알 수 있게 된다.
- **line 34~36**: case -1은 fork()가 제대로 실행되지 못했을 경우의 예외 처리를 나타낸다.