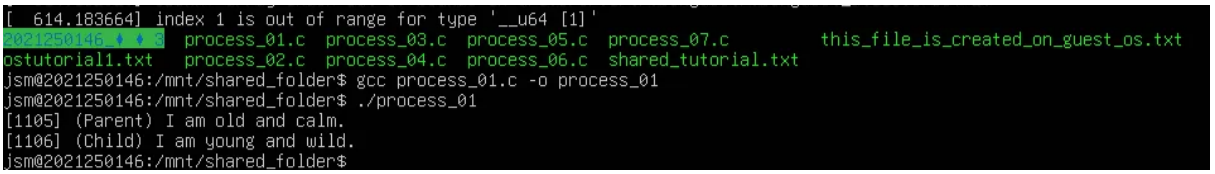
실습 과제3

2021250146 바이오시스템의과학부 정수민

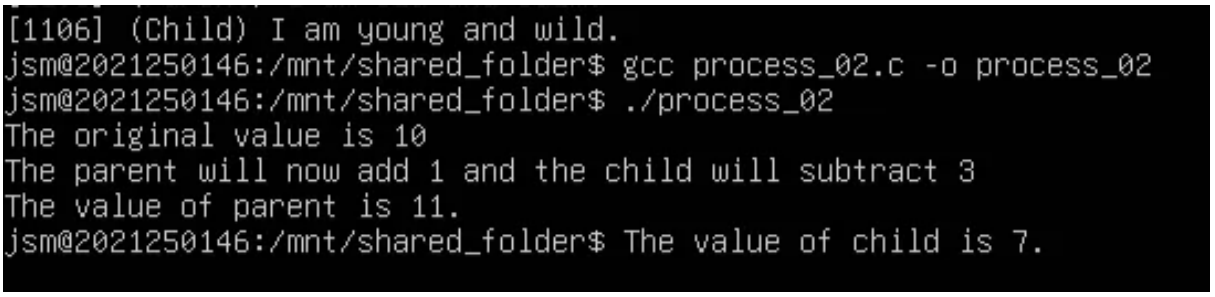
process_01

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



process_02

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



process_03

</1>: "/bin/pwd", "pwd", NULL

```
The value of parent is 11.
jsm@2021250146:/mnt/shared_folder$ The value of child is 7.

jsm@2021250146:/mnt/shared_folder$ gcc process_03.c -o process_03
jsm@2021250146:/mnt/shared_folder$ ./process_03
where am I?
/mnt/shared_folder
jsm@2021250146:/mnt/shared_folder$
```

process_04

빈칸	코드
</1>	default
</2>	case 0
</3>	pid2 = fork()
</4>	child_name, name, getpid()
</5>	case 0
</6>	grandchild_name, child_name, name, getpid()

```
where am I?
/mnt/shared_folder
jsm@2021250146:/mnt/shared_folder$ gcc process_04.c -o process_04
jsm@2021250146:/mnt/shared_folder$ ./process_04
My name is Jeffrey and I am a parent.
My name is Michael and my father is Jeffrey. My pid is 1126
My name is Steven! My father's name is Michael and my grandpa is called Jeffrey. My pid is 1127
jsm@2021250146:/mnt/shared_folder$
```

process_05

빈칸	코드
</1>	pid > 0
</2>	&statuswait()
</3>	3
</4>	pid == 0
</5>	0

```
My name is Steven! My father's name is Michael and my grandpa is called Jeffrey. My pid is 1127
jsm@2021250146:/mnt/shared_folder$ gcc process_05.c -o process_05
jsm@2021250146:/mnt/shared_folder$ ./process_05
Counting to three.
1!
2!
3!
jsm@2021250146:/mnt/shared_folder$
```

process_06

빈칸	코드	설명
</1>	`pid > 0`	첫 번째 `fork()` 의 부모 (parent)
</2>	`pid2 > 0`	두 번째 `fork()` 의 부모
</3>	`pid, &status, 0`	첫 번째 자식이 끝날 때까지 기다림
</4>	`pid2, &status, 0`	두 번째 자식이 끝날 때까지 기다림
</5>	`pid2 == 0`	두 번째 자식 (Child 2) 조건
</6>	`pid == 0`	첫 번째 자식 (Child 1) 조건

```
jsm@2021250146:/mnt/shared_folder$ gcc process_06.c -o process_06
jsm@2021250146:/mnt/shared_folder$ ./process_06
counting to 5!
1!
2!
3!
4!
5!
jsm@2021250146:/mnt/shared_folder$
```

process_07

process > code > C process_07.c > main(int, char * [])

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.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();

    //you are allowed to write your own printf
    switch(pid) {
        default:
            //hint: parent_pid =
            parent_pid=getpid();
            printf("[%d] I am a parent\n", parent_pid);
            //hint: how would this process get the result of another???
            wait(&status);
            if (WIFEXITED(status)) {
                int child_number = WEXITSTATUS(status);
                printf("[%d] ....and my child's favorite number is %d\n", parent_pid, child_number);
            }
            break;
        case 0:
            favorite_number = 5;
            //hint: child_pid =
            child_pid=getpid();
            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;
}
```

- 코드 설명
  - line 9-15
    - `pid` : `fork()` 반환값 저장 (부모/자식 식별)
    - `status` : `wait()`에서 자식 종료 상태를 저장
    - `parent_pid` , `child_pid` : 각각의 PID 저장용 변수
    - `favorite_number` : 자식이 좋아하는 숫자 저장
    - `favorite_fruit` : 부모가 좋아하는 과일 문자열 저장
    - `printf("Final Question!\n")` : 문제 안내 메세지 출력
  - line 17-20
    - `pid=fork()` : 자식 프로세스 생성 시스템 콜

      → 부모에게는 자식의 pid 반환

      → 자식에게는 0 반환

      → 실패하면 -1 반환
    - `switch(pid)` : fork()의 반환값에 따라 분기

      → `default` : 부모 프로세스

      → `case 0` : 자식 프로세스

      → `case -1` : fork 실패 시
  - line 21-31 (`default` : 부모 프로세스)
    - `parent_pid=getpid()` : 현재 프로세스의 PID를 getpid()로 구해 저장
    - `printf("[%d] I am a parent\n", parent_pid)` : 부모가 자기 PID와 함께 메세지 출력
    - `wait(&status)` : 자식 프로세스가 종료될 때까지 기다림, 종료 상태는 status 변수에 저장
    - `if (WIFEXITED(status)):` 자식이 정상 종료됐는지 확인
    - `int child_number = WEXITSTATUS(status)` : 자식이 exit()으로 반환한 값 추출
    - `printf("[%d] ....and my child's favorite number is %d\n", parent_pid, child_number)` : 부모가 자식의 favorite number 출력
  - line 32-39 (`case 0` : 자식 프로세스)
    - `favorite_number = 5` : favorite number를 5로 설정

        - `child_pid=getpid()`: 자신의 PID 저장
        - `printf("[%d] and I am a child!\n", child_pid)`: 자식 PID와 함께 자기소개
        - `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)`: 부모의 과일 출력 및 부모는 자식의 favorite number를 모른다는 내용 출력
        - `exit(favorite_number)`: 자식 프로세스 종료 → 종료 코드로 favorite number를 넘김
    - line 40-42 (`case -1`: 실패): 프로세스 생성에 실패한 경우 출력
    - line 44: 프로그램 정상 종료