과목명	운영체제	분반	X	담당교수	김화성 교수님
학과	전자통신공학과	학번	2016707079	이름	하상천
과제명: H/W 1 - Process 과제					

1. 과제설명 (사용자 요구사항 기술: 과제에 대한 설명 및 목표)

3.12 Using either a UNIX or a Linux system, write a C program that forks a child process that ultimately becomes a zombie process. This zombie process must remain in the system for at least 10 seconds. Process states can be obtained from the command

```
ps -l
```

The process states are shown below the S column; processes with a state of Z are zombies. The process identifier (pid) of the child process is listed in the PID column, and that of the parent is listed in the PPID column.

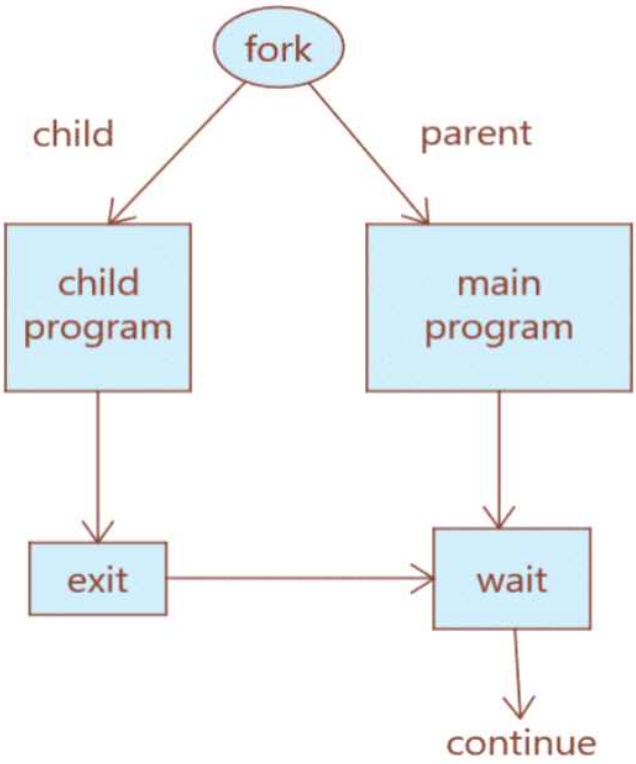
Perhaps the easiest way to determine that the child process is indeed a zombie is to run the program that you have written in the background (using the &) and then run the command `ps -l` to determine whether the child is a zombie process. Because you do not want too many zombie processes existing in the system, you will need to remove the one that you have created. The easiest way to do that is to terminate the parent process using the `kill` command. For example, if the process id of the parent is 4884, you would enter

```
kill -9 4884
```

좀비 프로세스를 만들고, ps -l 명령어를 통해 좀비 프로세스를 확인한다. 그리고 kill 명령어를 통해 부모 프로세스를 종료시켜 좀비 프로세스를 없앤다.

2. 사용자 요구사항을 정형적 방법으로 기술 (UML, Pseudo code, 그림등을 이용하여 기술)

3. 알고리즘 및 자료구조 설계 내용



4. 소스코드 설명 (직접 작성한 소스코드중에 핵심 부분을 발췌하여 설명)

```
#define ERROR -1
#define FINISH 0
```

가독성을 좋게 하기 위해 ERROR를 -1, FINISH를 0으로 치환하였다.

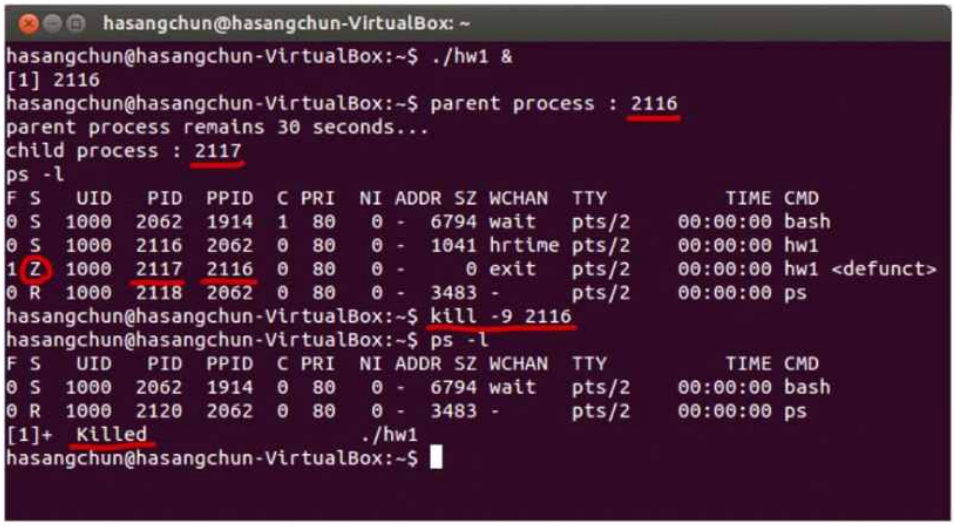
```
pid = fork();
        if(!pid){ /*  child process  */
                printf("child process : %d₩n", getpid());
                exit(FINISH);
        }
```

fork() 함수를 통해 자식 프로세스를 생성한다. fork() 함수가 성공하면 자식 프로세스는 리턴값으로 0을 받기 때문에 해당 조건문에 해당됨을 알 수 있다. 또한 getpid() 함수를 통해 PID(process ID) 값을 알 수 있다.

```
else{   /*  parent process  */
                printf("parent process : %d₩n", getpid());
                printf("parent process remains 30 seconds...₩n");
                sleep(30);
        }
```

부모 프로세스의 경우 자식 프로세스를 wait 하지 않기 때문에, 자식 프로세스는 좀비 프로세스가 된다. 특히, sleep 함수로 부모 프로세스가 30초 동안 잠들어있는 동안 좀비 프로세스가 된다.

5. 실행결과 및 설명 (실행 결과를 캡쳐하여 첨부한 후 설명)



(그림을 문서에 포함, 글자처럼 취급 옵션, 잉크 절약과 잘 보이게 하기위해 그림 반전)

작성된 프로그램을 실행한 결과 위와 같은 화면이 출력 되었다.

6. 고찰 (과제를 진행하면서 배운점 이나, 시행 착오 내용, 기타 느낀점)

이번 과제는 수업시간에 배운 프로세스에 관한 내용이었다. 방학동안 리눅스로 소켓통신을 하는 프로젝트를 진행했던 것이 도움이 많이 된 것 같다. fork() 함수를 통해 자식프로세스를 생성하고, 부모 프로세스는 sleep(30)을 통해 30초 남아있다가 프로그램이 종료되도록 하였다. 자식 프로세스는 부모 프로세스과 wait 하고 있지 않기 때문에 30초 동안 좀비 프로세스가 된다. 실험결과 사진을 보면, 부모 프로세스의 PID 값인 PPID가 2116이고, 자식 프로세스의 PID가 2117 임을 알 수 있다. 또한 S column 이 Z로 Zombie 프로세스임을 확인할 수 있다. 좀비 프로세스를 제거하는 방법 중 가장 쉬운 방법인 부모 프로세스를 kill 명령어를 통해 제거하였다. 인터넷에 검색해본 결과 -9는 강제종료이고, -15는 작업종료임을 알 수 있었다. kill -9 2116 명령어를 통해 부모 프로세스를 제거하고, 다시 ps -l 명령어로 프로세스의 상태를 확인해보니 좀비 프로세스가 제거되어있음을 확인 할 수 있었다.

처음에 hw1 실행파일을 실행할 때 ./hw1 & 으로 실행하였다. 인터넷에 검색해본 결과 &은 앞의 명령어를 백그라운드로 돌리고 동시에 뒤의 명령어를 실행하는 것이었다. &&은 앞의 명령어가 성공했을 때 다음 명령어가 실행되는 것이었다. ;은 앞의 명령어가 실패해도 다음 명령어가 실행되는 것이었다. 처음에는 하나도 몰랐지만 검색해보고 직접 코딩해봄으로서 더 오래 기억할 수 있을 것 같다. 뒤에 &을 붙이지 않고 ./hw1 으로 실행 했을 때는, 다른 터미널에서 ps -ef 명령어를 통해 좀비 프로세스 뿐만 아니라 실행중인 모든 프로세스의 정보들을 알 수 있었다.

7. 전체 소스코드 (글자크기 9에 줄간격을 120%로 유지하고 한 줄이 너무 길지 않게 작성)

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#define ERROR -1
#define FINISH 0
void main(){
        pid_t pid;
        pid = fork();
        if(!pid){ /*  child process  */
                printf("child process : %d₩n", getpid());
                exit(FINISH);
        }
```

```
        else if(pid < 0){          /*  error  */
                printf("Fork failed₩n");
                return ERROR;
        }
        else{   /*  parent process  */
                printf("parent process : %d₩n", getpid());
                printf("parent process remains 30 seconds...₩n");
                sleep(30);
        }
}
```

(글자크기는 10으로 유지하고 줄간격도 160%를 유지할 것)