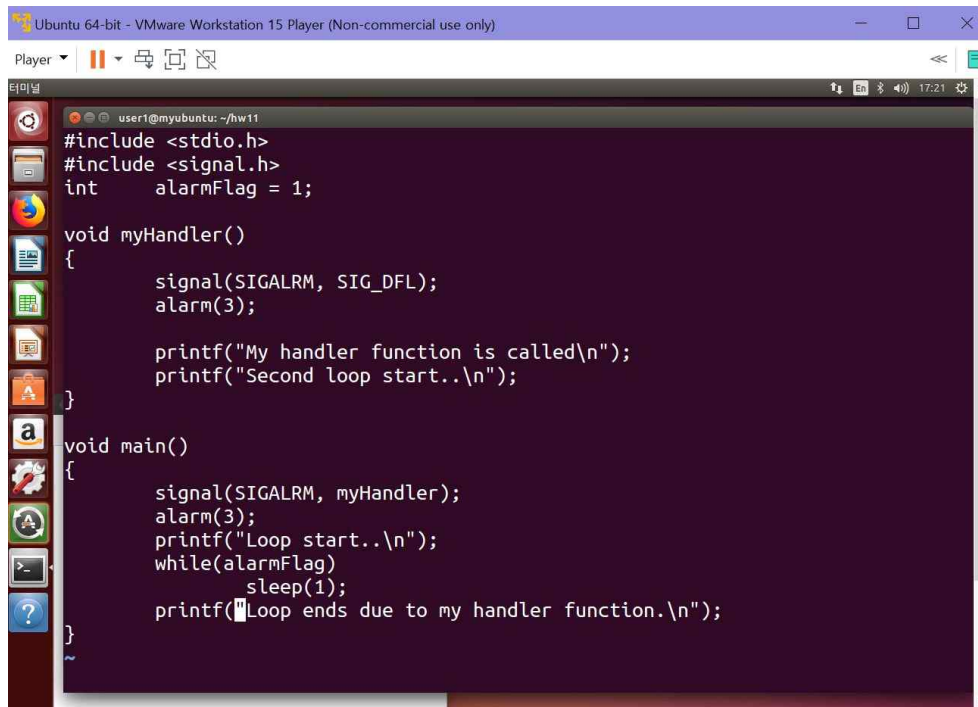


<시스템프로그래밍 - 실습13주차>

컴퓨터학과 20180976 송현수

- 실습1



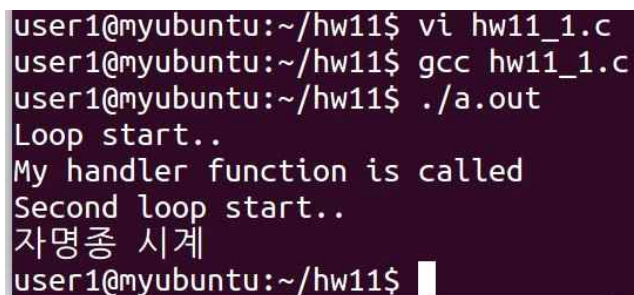
```
user1@myubuntu: ~/hw11
#include <stdio.h>
#include <signal.h>
int alarmFlag = 1;

void myHandler()
{
    signal(SIGALRM, SIG_DFL);
    alarm(3);

    printf("My handler function is called\n");
    printf("Second loop start..\n");
}

void main()
{
    signal(SIGALRM, myHandler);
    alarm(3);
    printf("Loop start..\n");
    while(alarmFlag)
        sleep(1);
    printf("Loop ends due to my handler function.\n");
}
```

=> 결과 화면



```
user1@myubuntu:~/hw11$ vi hw11_1.c
user1@myubuntu:~/hw11$ gcc hw11_1.c
user1@myubuntu:~/hw11$ ./a.out
Loop start..
My handler function is called
Second loop start..
자명종 시계
user1@myubuntu:~/hw11$
```

- 코드 설명

1. ./a.out을 실행하면, main()이 실행되어 "Loop start.."가 출력되고 무한루프를 돌다가 3초후 알람이 울리면 signal 함수에 의해 myHandler 함수로 넘어가게 된다.
2. myHandler 함수의 출력문들을 실행하고 alarmFlag = 0으로 설정하지 않았기 때문에 main()의 무한루프가 지속된다.
3. myHandler 함수가 호출된 후 3초후 울리는 alarm을 디폴트 값으로 변경했기 때문에 자명종 소리가 출력된 후 프로그램이 종료하게 된다.

- 실습2

```
user1@myubuntu: ~/hw11
#include <stdio.h>
#include <unistd.h>

void main()
{
    int fd[2], fork_return, childPID, status;
    char buf[32];

    pipe(fd);
    fork_return = fork();

    if(fork_return == 0){
        strcpy(buf, "Are you hyunsoo Song?");
        write(fd[1], buf, 32);
        exit(42);
    }
    else{
        childPID = wait(&status);
        read(fd[0], buf, 32);
        printf("Received message from child: %s\n", buf);
        printf("ChildPID : %d terminated with exit code %d\n", childPID,
status >> 8);
    }
}
"hw11_2.c" 27 lines, 514 characters

printf("The process with PID = %d terminates.\n", getpid());
}
```

=> 결과 화면

```
user1@myubuntu:~/hw11$ ./hw11_2.out
Received message from child: Are you hyunsoo Song?
ChildPID : 3908 terminated with exit code 42
The process with PID = 3907 terminates.
```

- 코드 설명

1. fork() return 값이 0이 아니면 부모 프로세스인데, 부모 프로세스는 wait() 함수로 자식 프로세스가 종료 될 때까지 기다린다.(동기화로 사용)
 2. 자식 프로세스는 연결된 파이프에 작성할 문장을 저장하고, exit() 함수로 자식 프로세스를 종료한다.
 3. 자식 프로세스가 종료되었으니 부모 프로세스는 파이프에 저장된 문장을 읽어서 출력한다. wait()로 반환된 자식 프로세스의 PID와 종료 반환 값도 출력해본다.
 4. 마지막 출력문으로 부모 프로세스 PID값을 출력한다.
- (fd[0] : read, fd[1] : write)

- 실습3

```
#include <stdio.h>

void main()
{
    printf("Before the first system()\n");
    system("echo the first system function");
    printf("After the first system()\n");

    printf("Before the second system()\n");
    system("sleep 100 &");
    printf("After the second system()\n");
    system("ps -ef | grep sleep");
    sleep(10);
    printf("Program end: hyunsoo Song\n");
}
```

=> 실행 결과

```
user1@myubuntu:~/hw11$ ./hw11_3.out
Before the first system()
the first system function
After the first system()
Before the second system()
After the second system()
user1      4005    1750  0 19:35 pts/0    00:00:00 sleep 100
user1      4006    4002  0 19:35 pts/0    00:00:00 sh -c ps -ef | grep sleep
user1      4008    4006  0 19:35 pts/0    00:00:00 grep sleep
Program end: hyunsoo Song
```

- 코드 설명

✓ system() 함수로 코드 내에서 셸 명령어를 사용하고 싶을 때 사용한다.

1. system()으로 echo 명령어 실행
2. system()으로 백그라운드에서 sleep 100 실행
3. system()으로 sleep이 들어간 프로세스 검색하여 sleep 100이 백그라운드로 실행되고 있음을 확인한다.

