

TI DSP, MCU 및 Xilinx Zynq FPGA 프로그램 전문가 과정

강사 - Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 - 은태영

zero_bird@naver.com

signal()

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <signal.h>
#include <stdio.h>

void my_sig(int signo)
{
    printf("my_sig called\n");
}

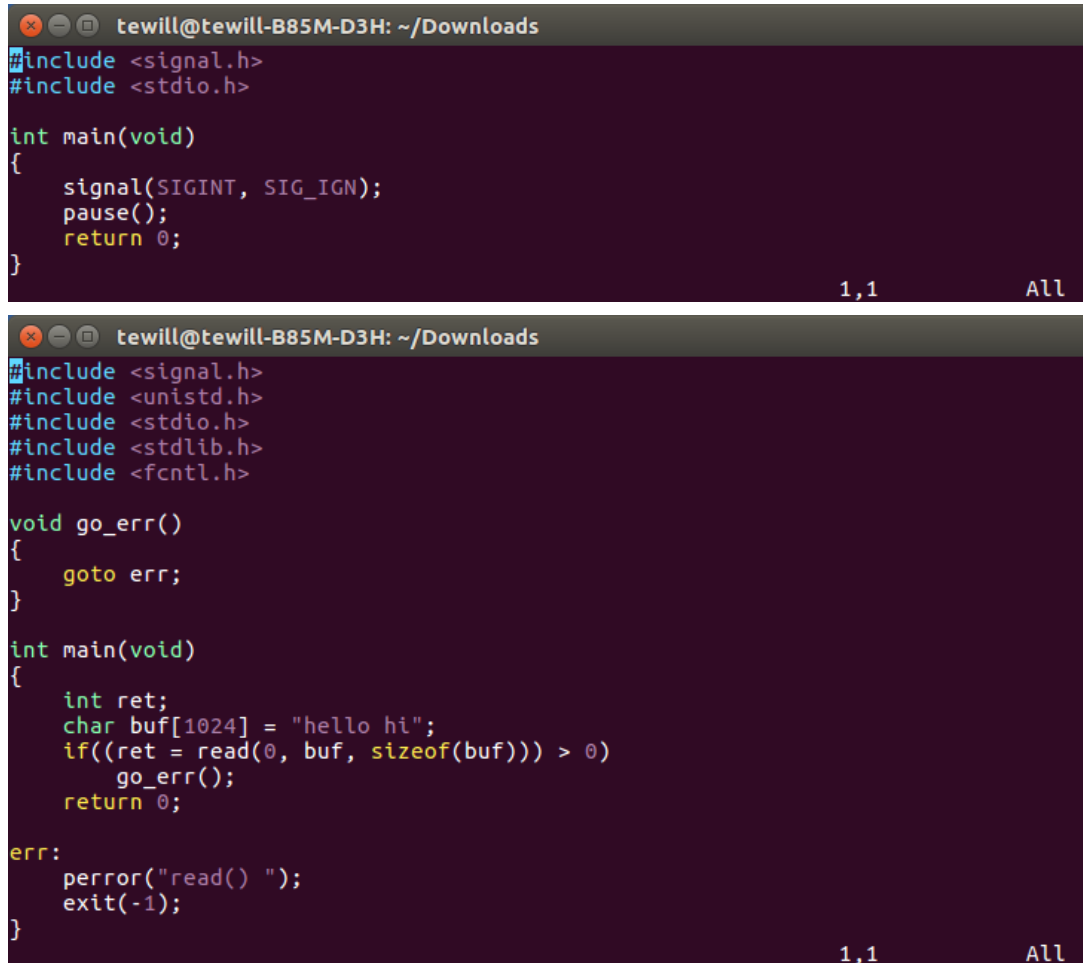
void my_sig2(int signo)
{
    printf("my_sig2 called\n");
}

int main(void)
{
    void (*old_p)(int);
    void (*old_p2)(int);
    old_p = signal(SIGINT, my_sig);
    pause();
    old_p2 = signal(SIGINT, my_sig2);
    pause();
    old_p2 = signal(SIGINT, old_p2);
    pause();
    for(;;)
        pause();
    return 0;
}
```

1,1 All

- ❖ signal() : 시그널의 리턴 값은 한단계 전에 등록했던 것을 리턴 한다.
- ❖ 즉 old_p 는 NULL 이 들어가 있고, old_p2 는 my_sig 가 저장되어 있다.

signal() and goto



The image shows two terminal windows. The top window displays a C program that includes `<signal.h>` and `<stdio.h>`. The `main` function calls `signal(SIGINT, SIG_IGN)` to ignore the SIGINT signal, then calls `pause()` to wait for any signal, and finally returns 0. The bottom window displays a C program that includes `<signal.h>`, `<unistd.h>`, `<stdio.h>`, `<stdlib.h>`, and `<fcntl.h>`. It defines a `go_err` function that uses `goto err;` to jump to an error handling section. The `main` function reads from the standard input into a buffer, and if the read operation fails, it calls `go_err()`. The `err` section prints an error message using `perror` and exits with a status of -1.

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <signal.h>
#include <stdio.h>

int main(void)
{
    signal(SIGINT, SIG_IGN);
    pause();
    return 0;
}
```

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <signal.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

void go_err()
{
    goto err;
}

int main(void)
{
    int ret;
    char buf[1024] = "hello hi";
    if((ret = read(0, buf, sizeof(buf))) > 0)
        go_err();
    return 0;
}

err:
    perror("read() ");
    exit(-1);
}
```

- ❖ SIGINT 를 SIG_IGN 한다.
- ❖ ctrl + c 가 입력 되도 무시한다.
- ❖ goto 는 stack 을 해제할 능력이 없다.
- ❖ 그렇기 때문에, 해당 코드는 동작하지 않는다.
- ❖ `perror("() ")` : 안에 들어있는 시스템 콜에 대한 결과 정보를 출력해 준다.

setjmp()

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <fcntl.h>
#include <stdlib.h>
#include <setjmp.h>
#include <stdio.h>

jmp_buf env;

void test(void)
{
    int flag = -1;
    if(flag < 0)
        longjmp(env, 1);
    printf("call test\n");
}

int main(void)
{
    int ret;
    if((ret = setjmp(env)) == 0)
        test();
    else if(ret > 0)
        printf("error\n");
    return 0;
}
```

13,26 All

- ❖ goto 를 해결하기 위해 setjmp 가 사용된다.
- ❖ setjmp() : goto 의 레이블을 설정하는 역할이다.
- ❖ 초기 리턴 값은 0 이며, 그 후 리턴 값을 조절할 수 있다.
- ❖ jmp_buf : 레이블의 위치이다.
- ❖ longjmp(jmp_buf, int) : jmp_buf 위치로 이동한다. 이때 setjmp 의 리턴 값을 int 값으로 한다.
- ❖ 따라서 call test 의 출력 없이 error 출력이 된다.

setjmp()

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <setjmp.h>

jmp_buf env;
jmp_buf env2;

void test(int i)
{
    if(i == 0)
        longjmp(env, 1);
    else if (i == 1)
        longjmp(env2, 2);
    else
        printf("error\n");
}

int main(void)
{
    int ret, ret2;
    if((ret = setjmp(env)) == 0) {
        printf("this\n");
        test(ret);
    }
    else if(ret > 0) {
        if((ret2 = setjmp(env2)) == 0)
        {
            printf("this env2\n");
            test(ret);
        }
        printf("ret = [%d]\n", ret2);
    }
    return 0;
}
```

1,1 All

- ❖ setjmp 를 좀 더 응용한 방식이다.
- ❖ this 출력 후, if 문 이 참이 되어 env 로 이동한다.
- ❖ 리턴 값이 1이기 때문에 else if 로 이동한다.
- ❖ this env2 를 출력하고 test 함수를 실행한다.
- ❖ ret 값이 1이기 때문에 env2 로 이동한다.
- ❖ 리턴 값이 0 이 아니기 때문에 ret = %d 를 출력한다.

alarm()

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <stdio.h>
#include <signal.h>
#include <fcntl.h>
#include <stdlib.h>

void my_sig(int signo)
{
    printf("You must insert coin\n");
    exit(0);
}

int main(void)
{
    char buf[1024];
    int ret;
    signal(SIGALRM, my_sig);
    alarm(3);
    read(0, buf, sizeof(buf));
    alarm(0);
    return 0;
}
```

1,1 All

- ❖ alarm() : 알람 역할을 한다.
- ❖ SIGALRM : 알람을 받았을 때를 나타낸다.
- ❖ 3초 안에 입력을 안 할 경우 my_sig 가 실행된다.
- ❖ 입력이 되었을 경우 alarm(0) 을 통하여, alarm 을 종료한다.

Up and Down Game

```
tewill@tewill-B85M-D3H: ~/Downloads
int main(void)
{
    int cor;
    int max = 9;
    int chan = 3;

    cor = game_start(max, chan);
    dataRead(cor, chan);

    return 0;
}
```

66,1 Bot

- ❖ 정답을 저장할 cor 을 선언한다.
- ❖ 숫자의 최대값을 저장할 변수를 선언한다.
- ❖ 기회를 나타낼 chan 을 선언한다.

Up and Down Game

```
tewill@tewill-B85M-D3H: ~/Downloads
#include <stdio.h>
#include <time.h>
#include <signal.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int game_start(int max, int chan)
{
    int cor = 0;
    srand(time(NULL));

    cor = (rand() % max) + 1;

    printf("게임 UP and DOWN! [1] ~ [%d]\n", max);
    printf("기회는 [%d]회 !!\n", chan);

    return cor;
}

void my_sig(int signo)
{
    printf("\n시간 초과!\n");
    exit(0);
}
```

1,1 Top

- ❖ rand 을 통하여 최대 값 안에서 정답을 만든다.
- ❖ 게임 정보를 출력한다.
- ❖ 정답을 리턴한다.
- ❖ alarm 이 사용될 경우 시간 초과를 출력한다.
- ❖ 해당 프로세스를 종료한다.

Up and Down Game

```
tewill@tewill-B85M-D3H: ~/Downloads
void dataRead(int cor, int chan)
{
    char buf[1024];
    int data = 0;
    int i = 0;

    while(i < chan)
    {
        signal(SIGALRM, my_sig);
        alarm(2);
        read(0, buf, sizeof(buf));
        alarm(0);
        data = atoi(buf);
        if(data == cor){
            printf("\n승리 !\n");
            break;
        }
        else if(data > cor)
            printf("[%d] 보다 작다\n", data);
        else
            printf("[%d] 보다 크다\n", data);

        i++;
    }
    if(i == chan)
        printf("\nGAME OVER\n");
}
```

26,0-1 65%

- ❖ signal 을 선언하여 alarm 호출 시, my_sig 를 실행한다.
- ❖ alarm 설정 후 표준 입력을 통해 데이터를 받는다.
- ❖ 입력이 완료되면 alarm 을 종료한다.
- ❖ 입력 받은 문자열을 숫자로 바꾼다.
- ❖ 비교를 통하여 승리, 작다, 크다 를 출력한다.
- ❖ i 를 증가시켜, 실패 시 chan 만큼 반복한다.
- ❖ 기회를 다 사용하면 game over 를 출력한다.

