

TI DSP, MCU 및 Xilinx Zynq FPGA

프로그래밍 전문가 과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 하성용
accept0108@naver.com

24 일차

시그널 기본코드

signal.c

```
#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);
```

/*SIGINT 에 대한 기본행동은 종료이기때문에, 특별히 시그널을 제어하지않을 경우 프로그램은 종료
가된다. 여기서는 signal 함수를 이용해서 SIGINT 에 대해서 my_sig 라는 함수를 실행하도록함*/

```
    pause();
```

```
    old_p2=signal(SIGINT,my_sig2); mysig2
```

첫번째는 널값들어

//이전에 시그널함수가

한단계전에걸 등록해줌 // 그래서 old_p 전에 아무것도없어서. old_p2 를 넣으면 그전인 my_sig 가
계속출력

```
    pause();
```

```
    old_p2=signal(SIGINT,old_p2); //
```

old_p 에 아무것도없어서 종료

old_p2 로 바꾸면 안끝나게되는데

시그널이 무언가 리턴한단것,

```
    pause();
```

```
    for(;;)
```

```
        pause();
```

```
    return 0;
```

```
}
```

signal.c

```
#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); //포인터 old_p 선언
    void(*old_p2)(int); //포인터 old_p2 선언
    old_p=signal(SIGINT,my_sig); //SIGINT 라는 시그널을 받으면 my_sig 를 실행하고
                                //old_p 에 NULL 을 반환

    pause();
    old_p2=signal(SIGINT,my_sig2); //SIGINT 라는 시그널을 받으면 my_sig2 를
                                //실행하고 old_p2 에 my_sig 를반환

    pause();
    old_p2=signal(SIGINT,old_p2); //SIGINT 라는 시그널을 받으면 old_p2 를 실행하고
                                //old_p2 에 my_sig2 를 반환

    pause();
    for(;;)
        pause();
    return 0;
}

```

구동하고 ctrl+c 3 번
메시지 2 번나오고 종료

```

^Cmy_sig called
^Cmy_sig2 called
^C

```

oldp 를 old_p2 로 바꾼후

```

^Cmy_sig called
^Cmy_sig2 called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called

```

PID 찾아서 번호날리기

```

#include<signal.h>
#include<stdio.h>

```

```

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

```

signal5.c

작성완료후 아래의 Command 를 수행한다.

```

$ps -ef|grep a.out
$kill -2 pid

```

gcc 실행후 터미널 우클릭 open terminel

```

yong@yong-Z20NH-AS51B5U:~/li/7$ ps -ef |grep a.out
yong      3781   3750   0 10:21 pts/2      00:00:00 ./a.out
yong      3799   3782   0 10:21 pts/3      00:00:00 grep --color=auto a.out
yong@yong-Z20NH-AS51B5U:~/li/7$ kill -2 3781
yong@yong-Z20NH-AS51B5U:~/li/7$ ps -ef |grep a.out
yong      3781   3750   0 10:21 pts/2      00:00:00 ./a.out
yong      3801   3782   0 10:22 pts/3      00:00:00 grep --color=auto a.out
yong@yong-Z20NH-AS51B5U:~/li/7$ kill -9 3781
yong@yong-Z20NH-AS51B5U:~/li/7$ ps -ef |grep a.out
yong      3803   3782   0 10:22 pts/3      00:00:00 grep --color=auto a.out

yong@yong-Z20NH-AS51B5U:~/li/7$ ./a.out
Killed

```

goto 예제

버퍼에 아무거나 입력하고 0 보다 크다로바꾸면 강제로 goto 에뜨게하기

```

#include<signal.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>

int main(void)
{
    int ret;
    char buf[1024]=fewfewfw;
    if((ret=read(0,buf,sizeof(buf))) <0) //sizeof 가 크기를 파악
        goto err;

    return 0;

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

```

```

yong@yong-Z20NH-AS51B5U:~/li/7$ ./a.out
abs
read() : Success

```

함수호출은 곧 스택을 만듦
 그런데 goto 가 스택을 해제할능력이없음
 이걸되게하려면 setjmp 와 longjmp 를 써야함

setjmp.c

```

#include<fcntl.h>
#include<stdlib.h>
#include<setjmp.h>

jmp_buf env;

void test(void)
{
    longjmp(env, 1);//setjmp 의 리턴값이 1 이된다는소리
    /*jmp_buf 는 상태를 저장하는 버퍼 구조체

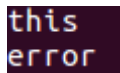
```

기본원리는 setjmp 는 현재 상태를 저장하고 longjmp 이전 상태로 이동하는 역할*/
}

```
int main(void)
{
    int ret; //ret 선언

    if((ret=setjmp(env))==0) //반환하는값을 ret 가 받음
    {
        printf("this\n");
        test();
    }

    else if(ret>0)
        printf("error\n");
    return 0;
}
```



```
this
error
```

실습
위 코드를 이용하여 함수에서 여러구간으로 뿔수있게 만들기

long jmp 와 setjmp 활용

강사님 코드
long_set_jmp.c

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <setjmp.h>

jmp_buf env1;
jmp_buf env2;

void test1(void)
{
    longjmp(env1, 1);
}

void test2(void)
{
    longjmp(env1, 2);
}

void test3(void)
{
    longjmp(env2, 1);
}

int main(void)
{
    int ret;
    if((ret = setjmp(env1)) == 0)
    {
        printf("this\n");
        test1();
    }
}
```

```

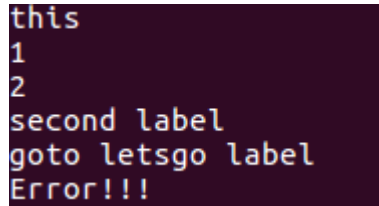
        else if(ret == 1)
        {
            printf("1\n");
            test2();
        }
        else if(ret == 2)
            printf("2\n");
        else
        {
            printf("goto letsgo label\n");
            goto letsgo;
        }

        if((ret = setjmp(env2)) == 0)
        {
            printf("second label\n");
            test3();
        }
        else
            longjmp(env1, 3);
letsgo:
            goto err; //에러로가

        return 0;

err:
        printf("Error!!!\n");
        exit(-1);
}

```



```

this
1
2
second label
goto letsgo label
Error!!!

```

setjmp2.c

```

#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;
}

```

error

long 점프 주의 : longjmp 로 가면 다시안돌아옴
long 점프 밑에 코드 적어도 무의미

setjmp3.c

```

#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;
}

```

랜덤으로 1~100 에 숫자를 받게해서 10 번의기회로 그걸 맞추는 게임 만들기

랜덤으로 숫자를 생성
1~100

50 넣어보면 업 다운 업
75 넣어보면 업 다운 다운
62 업 다운 업
68 OO

타이머는 1 초내로 안쓰면 지는게임

```

#include <time.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <stdbool.h>

void sig_handler(int signo)
{
    printf("You lose! Input should be within 1 second!\n");
    exit(0);
}

```

```

}

void make_game(int *data)
{
    *data = rand() % 100 + 1;
}

bool check_correct(int data, int cmp)
{
    if(data == cmp)
        return true; //같으면 참
    else
        return false; //다르면 거짓
}

void start_game(int data)
{
    char buf[32] = {0};
    bool fin;
    int i, cmp;

    for(i = 0; i < 10; i++)
    {
        signal(SIGALRM, sig_handler); //SIGALARM 받으면 시그 핸들러 시작
        printf("숫자를 맞춰봐!\n");
        alarm(1); //1 초안에 입력
        read(0, buf, sizeof(buf)); //
        alarm(0); //입력하면알람을 초기화시켜준다느거
        cmp = atoi(buf); //아스키투 인티저로 숫자로바꿔준거

        fin = check_correct(data, cmp); //이것이 맞는지 안맞는지

        if(fin)
        {
            printf("You Win!!!\n");
            exit(0);
        }
        else
        {
            if(data > cmp)
                printf("%d 보다 크다\n", cmp);
            else
                printf("%d 보다 작다\n", cmp);
        }
    }

    printf("You Lose!!! You Babo\n");
}

int main(void)
{
    int data;

    srand(time(NULL));
    make_game(&data);

    start_game(data);

    return 0;
}

```