TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 - 이상훈
gcccompil3r@gmail.com
학생 – 이우석
colre99@naver.com

[3/27(목) – 24 일차]

1 번

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

int main(void)
{
    signal(SIGINT, SIG_IGN); //  sig_ign는 막는다는뜻.
    signal(SIGQUIT, SIG_IGN); //종료 막음.
    signal(SIGKILL, SIG_IGN); //SIGKILL 은 못막음()
                              // ctrl + z 는 작성안해서..
   // 데몬을 끄려면,  kill-9번 하면 데몬도 끝남. 프로세스 자체를 죽임.
    pause();

    return 0;
}
```

결과값:

```
wooseok91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ps -ef | grep a.out
wooseok+  8413  5926  0 12:07 pts/2    00:00:00 ./a.out
wooseok+  8422  5926  0 12:09 pts/2    00:00:00 ./a.out
wooseok+ 10341  5926  0 13:14 pts/2    00:00:00 ./a.out
wooseok+ 13647  5926  0 18:54 pts/2    00:00:00 grep --color=auto a.out
wooseok91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ kill -9 8413
wooseok91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ps -ef | grep a.out
wooseok+  8422  5926  0 12:09 pts/2    00:00:00 ./a.out
wooseok+ 10341  5926  0 13:14 pts/2    00:00:00 ./a.out
wooseok+ 13651  5926  0 18:54 pts/2    00:00:00 grep --color=auto a.out
[1]   Killed                  ./a.out
```

2 번

```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);  //함수포인터
    void (*old_p2)(int); //2개 선언됨.
    old_p = signal(SIGINT, my_sig); // 시그널은 행동지침 설정. my_sig 실행.
    pause();
    old_p2 = signal(SIGINT, my_sig2); // my_sig2 실행
    pause();
    old_p2 = signal(SIGINT, old_p2); // old_p 에 아무것도 없다. old_p는 끝남.
    pause();                         // old_p2는 ctrl+c 해도 안끝남.
    for(;;)
       pause();

    return 0;
}                       /*시그널이 뭔가 리턴한다.
       시그널은 한단계 전에 저장했던걸 리턴해준다.
    즉, 첫번째 시그널은 값이 없고 ,두번째는 첫번째 시그널이 들어가고,
    세번째는 두번째께 들어간다.
                        */
```

결과값:

```
wooseok91@air:~/my
^Cmy_sig called
^Cmy_sig2 called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called
```

3 번

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

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

    return 0;
}

```

결과값:

```
wooseok91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ./a.out
asd
zxc
qwe
rty
vbn
```

4 번

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

void go(void)
{
    goto err;

}


int main(void)
{
    int ret;
    char buf[1024] = {'a', 'b'};
    if((ret = read(0, buf, sizeof(buf))) > 0)
      go();

    return 0;

    err:
       perror("read()"); //()안에 들어가있는 시스템콜이 어떤동작을
       exit(-1);          //                하는지 보여주는 함수 perror
}

//goto가 스택을 해제할 방법이 없다. 그래서 메인에 갈 방법이 없음
//되게 할 방법은 set점프랑 롱점프 로 해결.
```

5 번

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

jmp_buf env; //전역변수.

jmp_buf dnv;

int test(void)
{
   longjmp(env, 1);   // env 이라는 구간 알려줌. setjmp의 리턴값이 1이라는걸.
}

int tost(void)
{
   longjmp(dnv, 2);
}

int main(void)
{
   int ret;
   int let;

   if((ret = setjmp(env)) == 0)  // setjump가 있는 레이블이 goto가있는 레이블.
   {                             // 아무것도 세팅 안되있을시 처음엔 0. 조건만족.
      printf("this\n");     //롱점프에서 되돌아와 리턴값 1로 받아옴.
      test();
   }

   if((let = setjmp(dnv)) == 0)
   {
      printf("that\n");
      tost();
   }

```

```
38     else if(ret > 0)
39        printf("error0\n");
40
41
42     else if(let > 1)
43           printf("error1\n");
44
45
46     return 0;
47 }
48
49 //정리하자면 메인에 있는 if 문에서 셋점프가 세팅 안되어 있으니 값은 0
50 //조건 만족하니 밑으로 내려가 테스트로 간다. 가면 롱점프가 있고
51 // 롱점프 보기 env에 값을 1로 리턴해줌. 롱점프 역할 다 했으니 다시,
52 //메인에 있는 if문의 셋점프로 온다. 원래값은 0이었으나, 롱점프에서 1로 변함.
53 //이제 조건이 맞질 않으니 else if로 온다. 조건이 0보다크니, 조건만족하면서
54 //에러를 출력함.
55
56
57 //좋은예시 lec/lsp의 setjmp.c 열어보면 왔다갔다 하는걸 확인할 수 있다.
58 //  goto 와 setjump의 활용을 잘 볼 수 있다.
59
60 //setjump 와 goto가 같은 레이블인걸 확인 가능하고, goto 와 longjump 와 같은역할.
```

결과값:

```
wooseok91@air:~/my_proj/Homework/sanghoonlee/lec/lsp$ ./a.out
this
that
error0
```