Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정
#24

강사:Innova Lee(이 상훈)
학생: 김시윤

수업내용 복습

---signal---

```
#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 두개 시그날 은 행동지침 결정
	void (*old_p2)(int);
	old_p = signal(SIGINT,my_sig);//시그인트가 잡히면 마이시그
	pause();
	old_p2 = signal(SIGINT,my_sig2);
	pause();
	old_p2= signal(SIGINT,old_p);
	pause();
	for(;;)
		pause();
	return 0;
}
```

//old_p2 하면 안끝남 이유는 signal 이 먼가를 리턴해 동작시키면 마이시그가 먼저 나오고 그담에 마이시그 2 콜드가 나옴 시그날은 기본적으로 과거 한단계 전에 등록했던걸 리턴해준다.

//맨첨에 시그널 등록했을때 과거에 아무것도 없기때문에 널이 들어감 시그 2 에는 mysig 가 들어감.

//그래서 old_p2 가 들어가면 마이시그가 계속들어감.

---

void (*old_p)(int);
void (*) (int) old_p
old_p 라는 이름의 인트형 인자를 갖고 void 를 반환하는 함수포인터.
void (*old_p2)(int);
void (*) (int) old_p2
old_p2 라는 이름의 인트형 인자를 갖고 void 를 반환하는 함수포인터.
함수포인터는 함수의 주소를 담는다.
여기서 singnal 함수를 담기위해 사용하였다.

여기서 signal 함수에는 SIGINT 가 있다.

SIGHUP	터미널을 읽어버렸을때 발생한다.
SIGABRT	프로그램의 비정상종료시 발생한다.
SIGINT	Control-C 나 DELETE 키를 입력했을때 발생한다.
SIGIO	비동기적인 입출력이 발생했을때
SIGKILL	프로세스를 죽이기 위해서
SIGPIPE	단절된 파이프에 write 할경우 발생
SIGSEGV	잘못된 메모리 참조(주로 포인터를 잘못 썼을때)
SIGSTOP	프로세스의 일시중단 (Ctrl+z)
SIGSUSR1	사용자를 위해 정의된 시그널

여기서 SIGINT 는 Ctrl+c 또는 Del 키를 입력했을때 발생하는 시그날이다.
컨트롤 씨를 눌러 SIGINT 가 잡히면 mysig 로 간다. 그리고 sig1 called 를 출력시킨다.
그리고 다시 메인으로 돌아오고 정지시킨다.
그리고 두번째에서 SIGINT 가 또 발생하면 mysig2 함수로 가서 sig2called 를 출력시키고 다시 메인으로 돌아와서 정지한다.
그리고 한번 더 했을경우 old_p 는 반환값이 NULL 이기 때문에 동작을 하지 않고 종료하게된다 하지만 old_p 를 old_p2 로 바꾸었을경우 signal 함수가 old_p 를 반환하기 때문에 sig1_called 가 계속 할때마다 출력된다.

```
^C
siyun@siyun-Z20NH-AS51B5U:~/my_proj/class24$ ./a.out
^Cmy_sig called
^Cmy_sig2 called
^C
siyun@siyun-Z20NH-AS51B5U:~/my_proj/class24$ vi signal
siyun@siyun-Z20NH-AS51B5U:~/my_proj/class24$ vi signal2.c
siyun@siyun-Z20NH-AS51B5U:~/my_proj/class24$ vi signal.c
siyun@siyun-Z20NH-AS51B5U:~/my_proj/class24$ gcc signal.c
signal.c: In function ‘main’:
signal.c:19:2: warning: implicit declaration of function ‘p
  pause();
  ^
siyun@siyun-Z20NH-AS51B5U:~/my_proj/class24$ ./a.out
^Cmy_sig called
^Cmy_sig2 called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called
^Cmy_sig called
```

---signal---

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

int main(void)
{
	signal(SIGINT,SIG_IGN);
	pause();
	return 0;
}
//kill -2 는 시그인트
```

signal 신호는 프로세스가 취할 행동의 메뉴얼이다.
여기서 SIGINT 는 ctr+c 를 나 del 을 누르면 발생하는 시그날 신호이다.
이 신호을 무시했다.
그리고 실행시키면 프로세스가 ctr +c 를 해도 종료되지 않는다.
하지만 kill pid 하면 지워진다. (또는 kill -2 빼고 다 됨)
여기서 signal 함수에 SIGINT 에 2 를 넣어도 SIGINT 와 동일한 동작을 한다.
이유는 시그인트가 2 번 시그날 신호이기 때문이다.

---setjmp---

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

jmp_buf env;//env 점프버프로 변수 선언
//goto err; 에서 setjmp(env) env = err

void test(void)
{
	longjmp(env,1);//env 는 이프문에 env 라는 소리 리턴값 1 로 바꾼다는 소리.
}//1 대신 0 을 넣으면 무한으로 돈다.
```

```
int main(void)
{
	int ret;
	if((ret = setjmp(env))==0)//goto 의 레이블 셋점프 첨에 하면 무조건 0 err
	{
	printf("test\n");
	test();

	}
	else if(ret > 0)//다시 돌아와서 리턴값 1 이 되서 에러 그래서 error 출력
		printf("error\n");
	return 0;
}
```

setjmp(env) == goto env; 라고 보면된다.
setjmp 를 하면 처음에 무조건 0 이다. setjmp 가 0 이이면 test 함수를 가서 longjmp 가서 1 을 갖고 온다. 그래서 setjmp 가 반환하는 값은 1 이되고 그 값을 ret 에 저장한다.
ret 가 0 보다 커졌기 때문에 else 문을 통과하게 되고 에러라는 출력이 나온다.

---setjmp 레이블 추가 ---

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

jmp_buf env;
jmp_buf env2;

void test(void)
{
	longjmp(env,1);
}

void test2(void)
{
	longjmp(env2,1);
}

int main(void)
{	int a;
	int ret2;
	int ret;
	if((ret = setjmp(env))==0)
	{
		printf("test\n");
		test();

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

	if((ret2 = setjmp(env2))==0)
	{

		test2();
	}
	else if(ret2>0)
	{	a=2;
		printf("a= %d \n",a);
	}
	return 0;
}
```

```
#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;

	return 0;

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

위에는 내가 수업시간에 짠 소스코드고 아래는 선생님께서 짠 소스코드다.
비교해보면 내꺼는 너무 단순하다.
처음에 셋점프의 env1 의 반환값은 0 이다 그래서 이프문을 통과하여 this 를 프린트하고 테스트 1 함수로 들어간다.
테스트 1 을 들어가면 롱 점프로 env1 과 1 을 반환해준다. 그러면 셋점프의 env1 은 0 이 아니기 때문에 첫번째 이프문을 통과하지 못하고 밑에 ret ==1 을 통과하게되어 1 을 출력하고 테스트 2 함수로 간다. 테스트 2 함수에서 setjmp(env1)를 2 로 만들어주고 ret==2 이프문을 통과하게 된다 그럼 2 를 출력해준다.
그리고 setjmp(env2)가 0 인 이프문이 참이라 실행되고 세컨드레이블 을 출력하고 테스트 3 함수로가서 env2 를 1 로 만들어주고 이제 모든 이프문을 충족하지 않기에 엘지 고투 렛츠고 레이블을 출력하고 렛츠고로 가서 에러를 출력한다.

---alarm---

```
#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;
}
//3 초안에 입력 안하면 코인넣으래
```

---

alarm 은 초단위 후에 프로세스에 SIGALRM 을 전달한다 고 되어있다.
만약 초단위가 0 이라면 SIGALRM 이 전달되지 않을것이다.
여러개 쓰인다면 최근에 설정한 alarm 설정값으로 저장된다.
반환값 = alarm 은 이전에 설정된 알람이 시그널을 전달할 때까지 남은 시간을 초 단위 숫자로 반환하거나, 이전에 설정된 알람이 없을경우 0 을되돌려 준다.

---alarm 을 이용한 게임---

```
#include <stdio.h>
#include <signal.h>
#include <fcntl.h>
#include <stdlib.h>
#include <time.h>

void my_sig(int signo)
{
        printf("You loose\n");
        exit(0);
}
//alarm0 을 하면 signal 멈춤
void comp(int rnd)
{
        int cnt =1;
        int num;
        char buf[1024];

        for(;;)
        {
                signal(SIGALRM, my_sig);
        alarm(3);
                read(0,buf,sizeof(buf));
        num=atoi(buf);
        alarm(0);

                if(rnd == num)
        {
                printf("youwin\n");
                        break;
        }
        else if(cnt < 5)
        {
                if(rnd > num)
                {
                printf("%d 보다 더커 \n",num);
                cnt++;
                }
                else if(rnd < num)
                {
                        printf("rnd 는 더 작아 \n");
                  cnt++;

                }
        }
        else if(cnt >= 5)
```

```
                {
                        printf("you loose\n");
                        break;
                }

        }
}

int main(void)
{
        int rnd;

        srand(time(NULL));
        rnd = rand() % 100+1;
        comp(rnd);

return 0;
}
```