

TI DSP,Xilinx zynq FPGA,MCU 및
Xilinx
zynq FPGA 프로그래밍 전문가 과정

강사-INNOVA LEE(이상훈)

Gccompil3r@gmail.com

학생-윤지완

Yoonjw7894@naver.com

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<fcntl.h>

int main(void)
{

int fd,nread;
char buf[1024];

fd=open("yoon.txt",O_WRONLY|O_CREAT|O_TRUNC,0644);
nread=read(0,buf,sizeof(buf));
write(fd,buf,nread);
return 0;
}
```

```
#include<stdio.h>
#include<signal.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);//함수 포인터
void (*old_p2)(int);
```

```

old_p= signal(SIGINT, my_sig);//처음엔 NULL 값을 리턴
pause();
old_p2 = signal(SIGINT , my_sig2);
pause();

```

old_p2 = signal(SIGINT,old_p2);//한단계전에 등록된것을 리턴 시킨다.그러니까 my_sig2 전에 my_sig 가 들어가있기때문에 my_sig 가 계속 호출이 된다.

```

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

```

```

}

```

이렇게 코드를 짜면 컨트롤 c를 계속 이어진다 끝나지않는다.

```

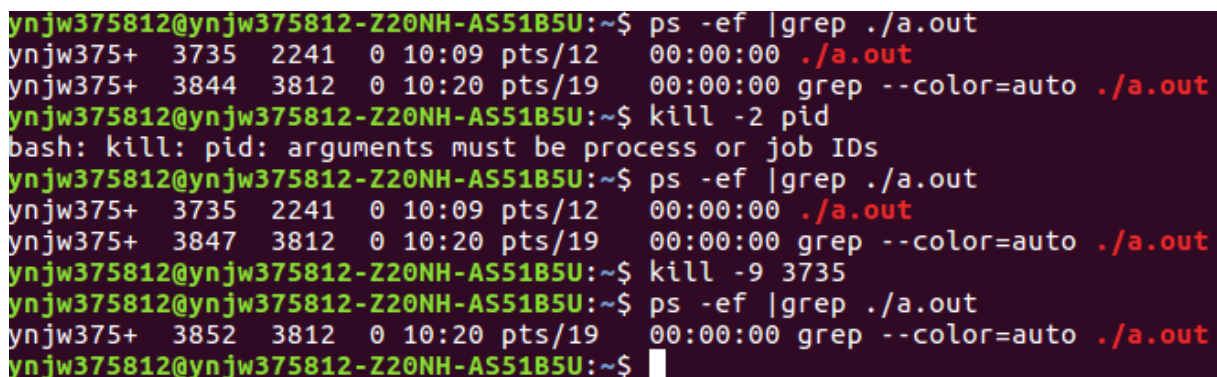
#include<stdio.h>
#include<signal.h>

```

```

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

```



```

ynjw375812@ynjw375812-Z20NH-AS51B5U:~$ ps -ef |grep ./a.out
ynjw375+  3735  2241  0 10:09 pts/12    00:00:00 ./a.out
ynjw375+  3844  3812  0 10:20 pts/19    00:00:00 grep --color=auto ./a.out
ynjw375812@ynjw375812-Z20NH-AS51B5U:~$ kill -2 pid
bash: kill: pid: arguments must be process or job IDs
ynjw375812@ynjw375812-Z20NH-AS51B5U:~$ ps -ef |grep ./a.out
ynjw375+  3735  2241  0 10:09 pts/12    00:00:00 ./a.out
ynjw375+  3847  3812  0 10:20 pts/19    00:00:00 grep --color=auto ./a.out
ynjw375812@ynjw375812-Z20NH-AS51B5U:~$ kill -9 3735
ynjw375812@ynjw375812-Z20NH-AS51B5U:~$ ps -ef |grep ./a.out
ynjw375+  3852  3812  0 10:20 pts/19    00:00:00 grep --color=auto ./a.out
ynjw375812@ynjw375812-Z20NH-AS51B5U:~$

```

kill -2 pid 는 SIGINT 를 지우는 번호이다. -2 로 했지만 지워지지않고 -9 3735 했을때 지워지는 것을 확인 할 수 있었다.

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

```
int main(void)
{
    int ret;
    char buf[1024]="yoon ji";
    if((ret = read(0,buf,sizeof(buf)))>0)
        goto err;
    return 0;
err:
    perror("read()");//이 시스템에 동작이 어떻게 되었나 보는것.
    exit(-1);
}
```

```
ynjw375812@ynjw375812-Z20NH-AS51B5U:~$ ./a.out
$
read(): Success
}
```

```
#include<stdlib.h>
#include<fcntl.h>
#include<unistd.h>
```

```
void success (char **buf)
{
char **tmp=buf;
*tmp="woon ji";
int ret;
if((ret = read(0,tmp,sizeof(tmp)))>0)
```

```
goto err;
}
```

```
int main(void)
```

```
{
char *buf[1024];
success(buf);
return 0;
err:
perror("read()");
exit(-1);
```

goto err;//goto 는 스택을 해제할수 있는 권한이 없기때문에 main 으로 다시
가지못하>는 것이다.

그래서 이것을 보완하는것이 다음 코드이다.

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

```
jmp_buf env;
```

```
void test (void )
```

```
{
```

```
longjmp(env,1);
```

```
}
```

```

int main(void)
{
int ret;
if((ret=setjmp(env))==0)//이 부분이 goto err 의 동작이다.
처음 리턴값은 무조건 0 이다.longjmp 하고 다시 이 부분으로 돌아오고 1 을
가지고 오고 ret=1 의 값을 가지므로 error 이 동작이 된다.
test();
else if(ret>0)
printf("error\n");
return 0;
}

```

```

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

```

```

jmp_buf env;
jmp_buf env1;

```

```

void test (void )
{

```

```

longjmp(env,1);
}
void test2(void)
{
longjmp(env,2);
}

```

```

int main(void)
{
int ret;

if((ret=setjmp(env))==0)
{
printf("this\n");

```

```

test();
}
else if(ret==1)
{
printf("error\n");
test2();

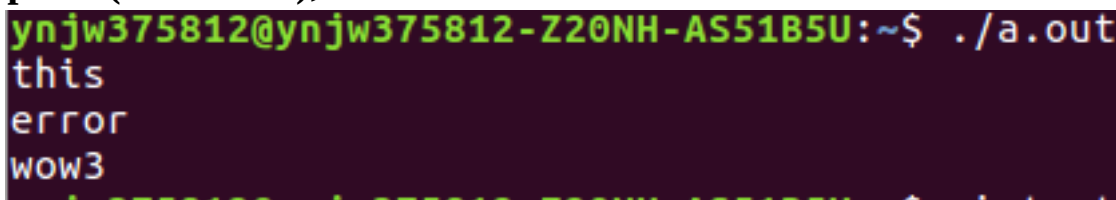
}

```

```

else if(ret>1)
{
printf("wow3\n");

```



```

ynjw375812@ynjw375812-Z20NH-AS51B5U:~$ ./a.out
this
error
wow3

```

```

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

```

```

jmp_buf env;

```

```

void test (void )
{
int flag = -1;
if(flag <0)
{

```

longjmp(env,1); //longjmp 는 그 아래 명어들을 수행하지 않으니 이점 주의
해야 한다.

```

printf("call test\n");
}
}

```

```

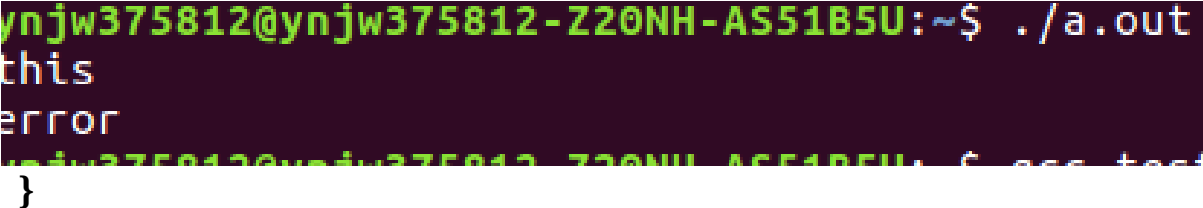
int main(void)
{

```

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

}

return 0;
```

A terminal window with a dark background and green text. The prompt is 'ynjw375812@ynjw375812-Z20NH-AS51B5U:~\$'. The user enters './a.out'. The output shows 'this' on the first line and 'error' on the second line. The prompt is visible again on the third line.

```
ynjw375812@ynjw375812-Z20NH-AS51B5U:~$ ./a.out
this
error
ynjw375812@ynjw375812-Z20NH-AS51B5U:~$
```

```
void my_sig (int signo )
{
printf("you fail\n");
exit(0);
}
```



```

#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;
else
return false;
}

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

for(i = 0; i < 10; i++)
{
signal(SIGALRM, sig_handler);
printf("숫자를 맞춰봐!\n");
alarm(1);
read(0, buf, sizeof(buf));
alarm(0);
cmp = atoi(buf);
fin = check_correct(data, cmp);
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;
}
```