CSED211: Lab. 11

Shell Lab 2

김보석
boseok@postech.ac.kr

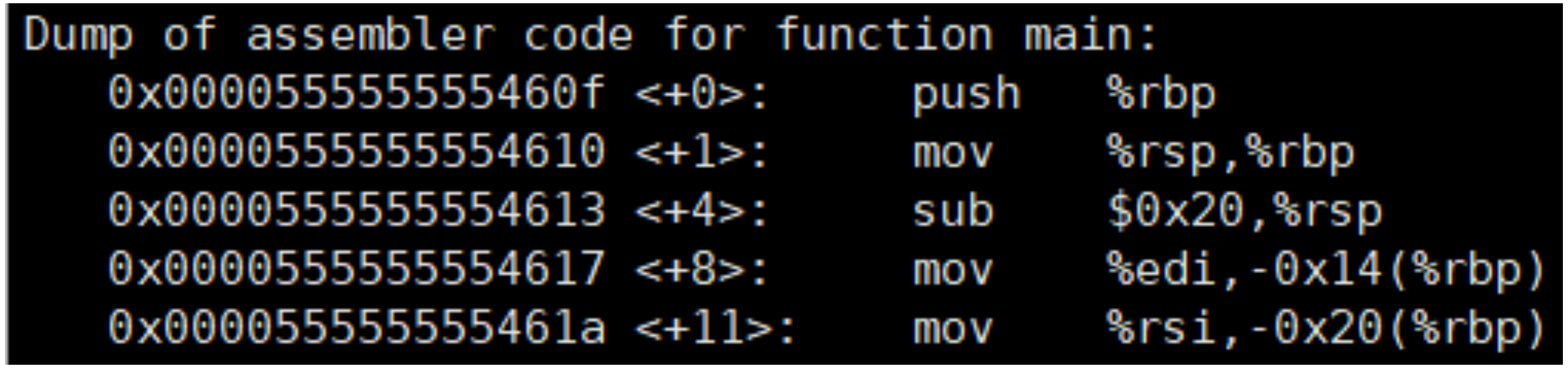
POSTECH

2023.12.04

Table of Contents

Control Flow

- Program Flow
  - **`Jump`**
    - **`Condition`**
    - **`Iteration`**
    - **`Branch`**
    - **`Call ~ Return`**



```
Dump of assembler code for function main:
   0x000055555555460f <+0>:     push   %rbp
   0x0000555555554610 <+1>:     mov    %rsp,%rbp
   0x0000555555554613 <+4>:     sub    $0x20,%rsp
   0x0000555555554617 <+8>:     mov    %edi,-0x14(%rbp)
   0x000055555555461a <+11>:    mov    %rsi,-0x20(%rbp)
```

Control Flow

- `Jump` cannot handle
  - Errors
    - Divide by zero
    - Out of memory
  - Asynchronous signals
    - Data arrives from hard drive
    - User wants to stop the program
    - …

Control Flow

- `Jump` cannot handle
  - Errors
    - Divide by zero
    - Out of memory
  - Asynchronous signals
    - Data arrives from hard drive
    - User wants to stop the program
    - …

“Exceptional Control Flow”

Exceptional Control Flow

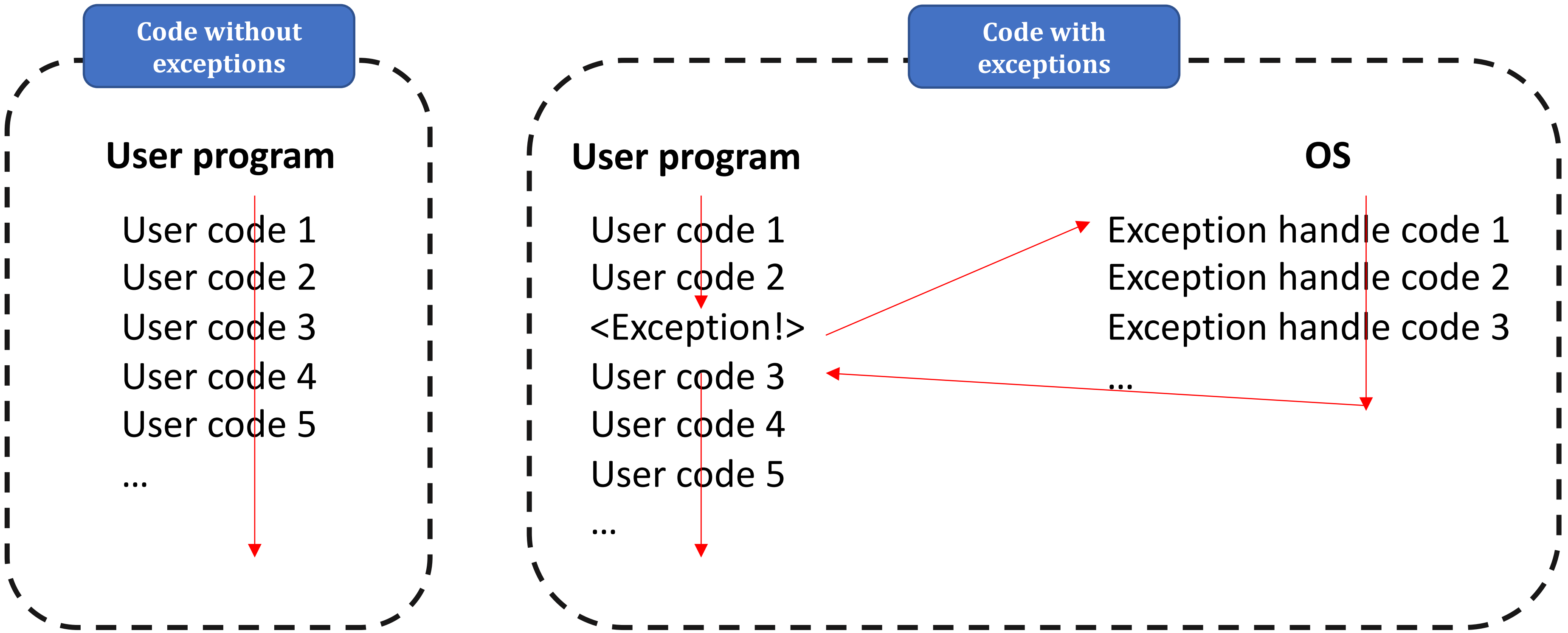
- Exceptions
- Process Context Switch
- Signals
- Nonlocal Jumps

Exceptional Control Flow

- Exceptions
- ~~Process Context Switch~~
- Signals
- ~~Nonlocal Jumps~~

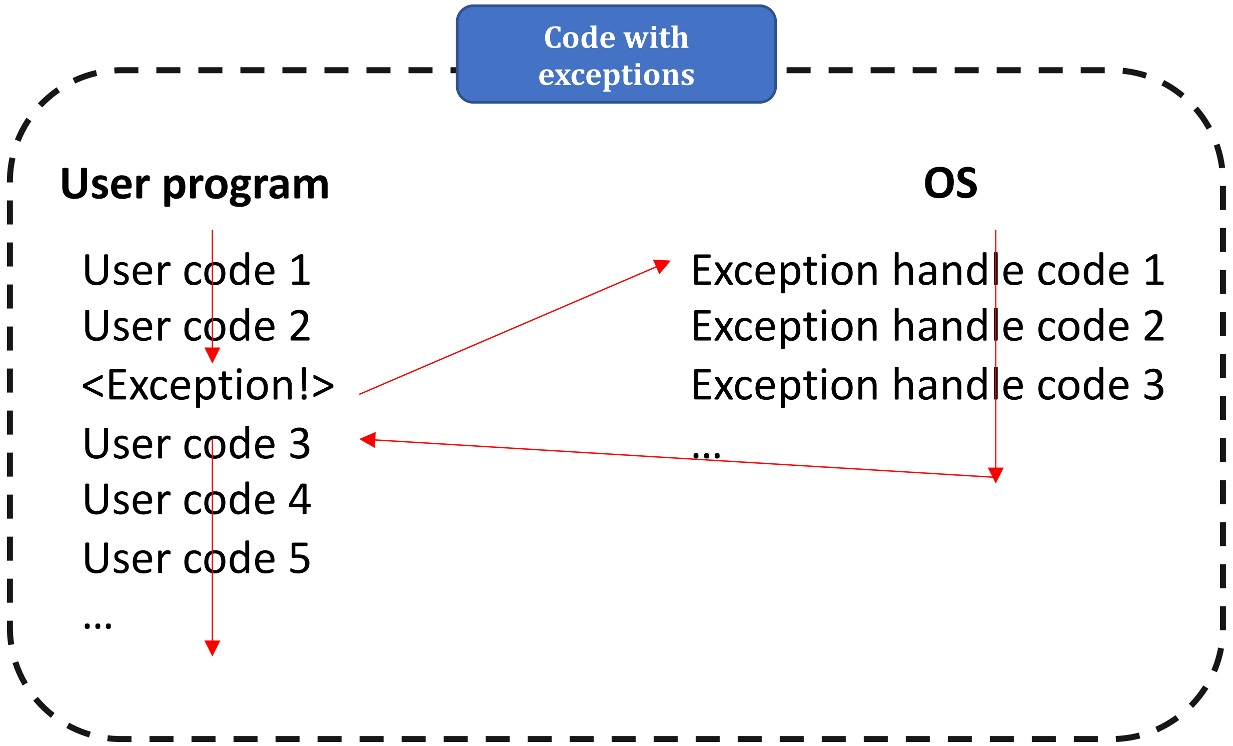
Exceptions

- OS deals the exceptions



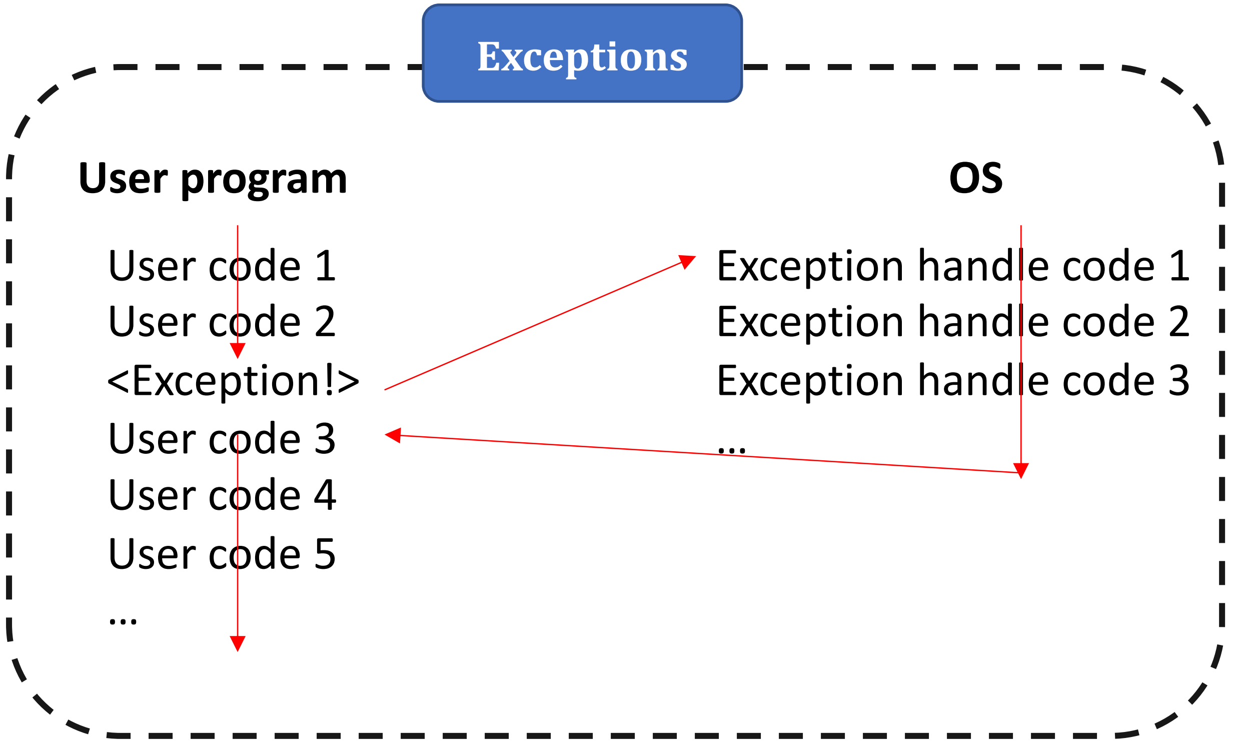
Exceptions

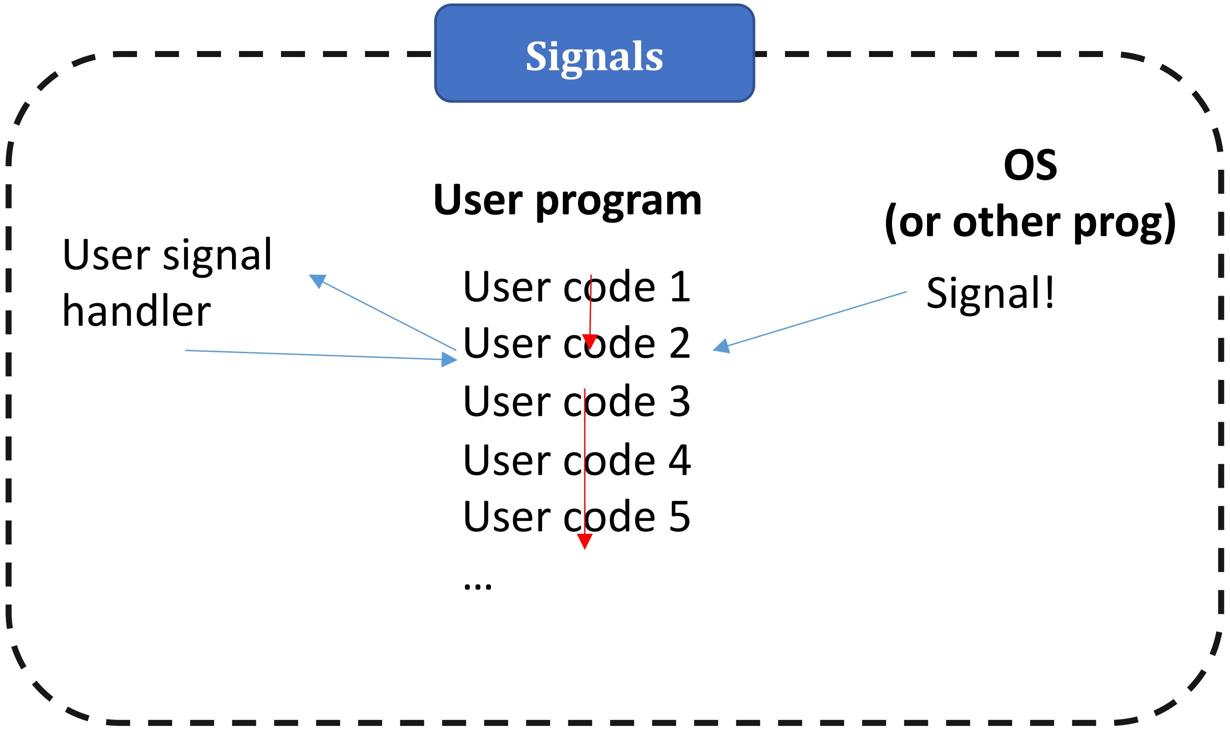
- Page fault
  - Memory related
- Ctrl + c
  - User input
- …



Signals

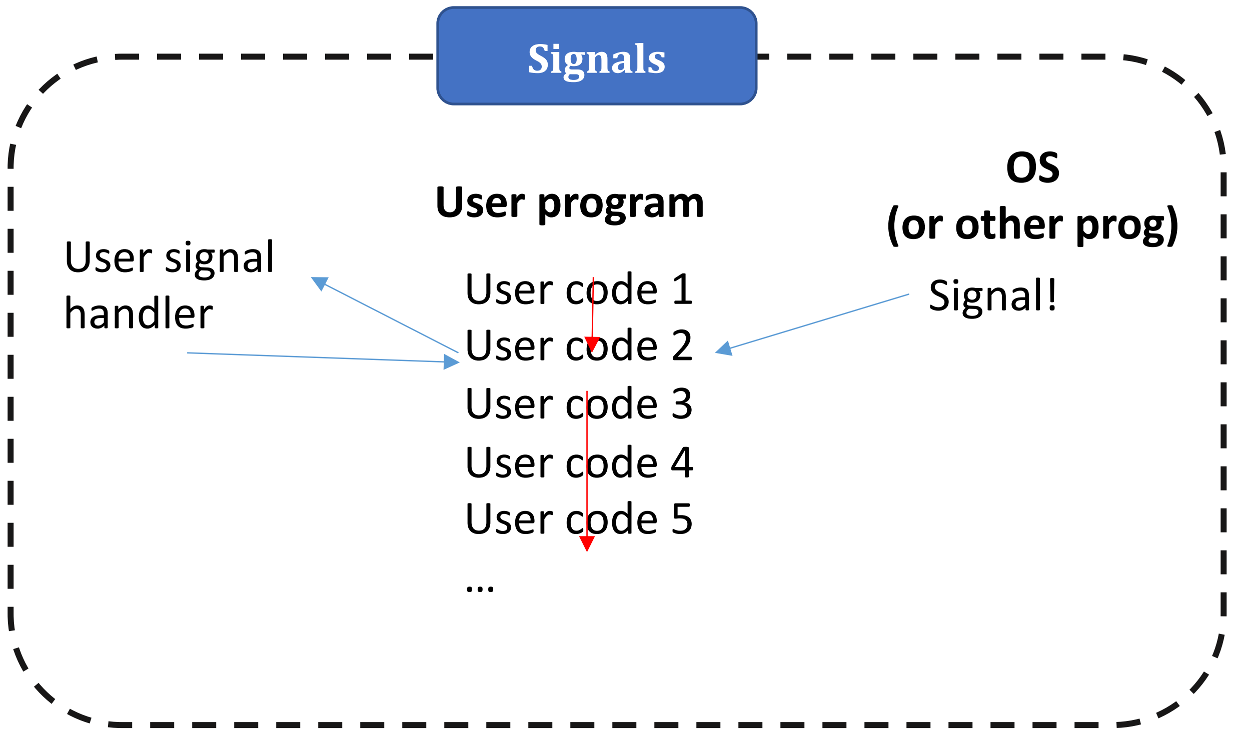
- A small message that notifies a process event of some type has occurred in the system
  - Similar to exceptions





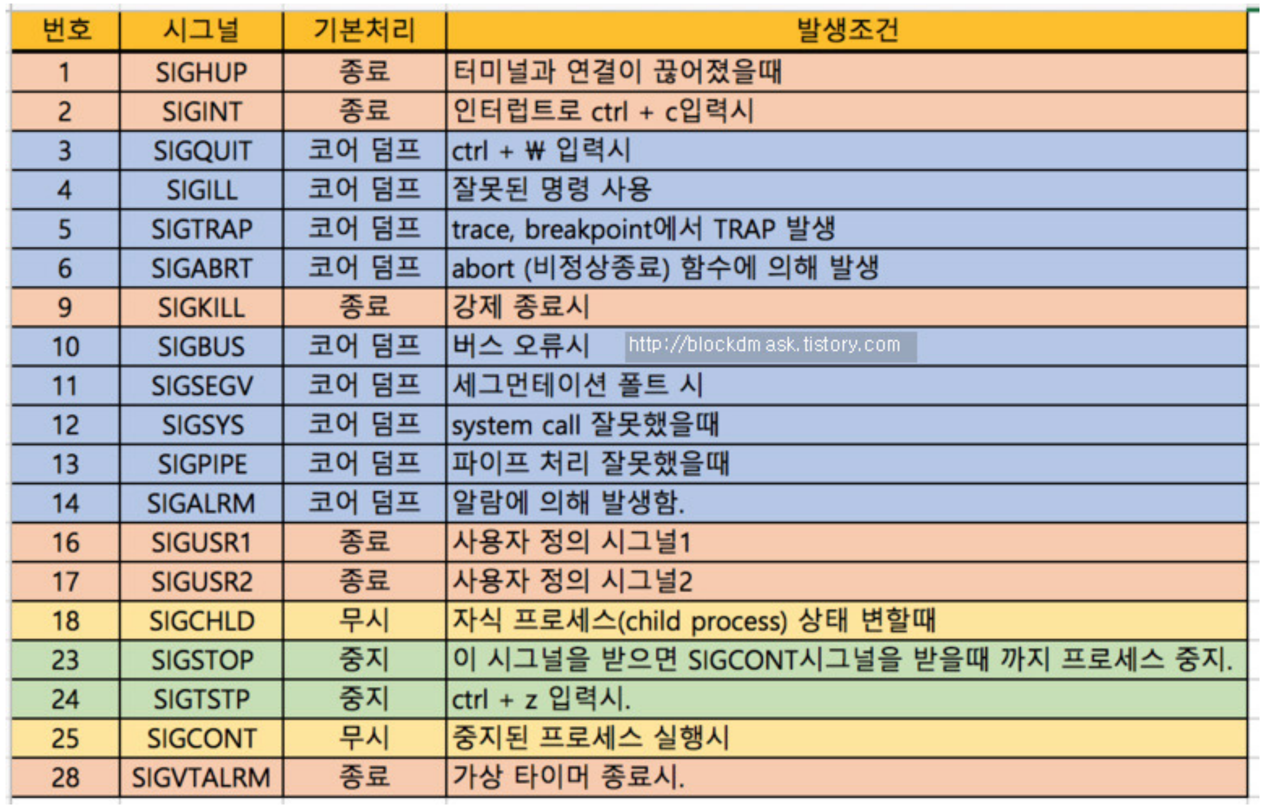
Signals

- Signal only contains
  - Signal ID
  - No detailed information



Signals

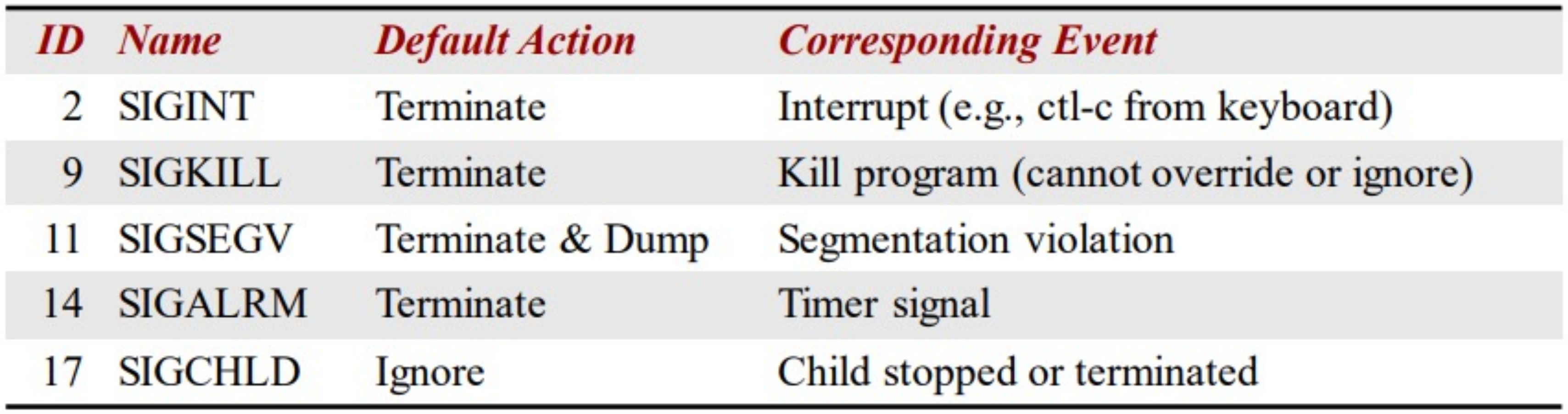
- Each signal ID represents a situation



번호	시그널	기본처리	발생조건
1	SIGHUP	종료	터미널과 연결이 끊어졌을때
2	SIGINT	종료	인터럽트로 ctrl + c입력시
3	SIGQUIT	코어 덤프	ctrl + ₩ 입력시
4	SIGILL	코어 덤프	잘못된 명령 사용
5	SIGTRAP	코어 덤프	trace, breakpoint에서 TRAP 발생
6	SIGABRT	코어 덤프	abort (비정상종료) 함수에 의해 발생
9	SIGKILL	종료	강제 종료시
10	SIGBUS	코어 덤프	버스 오류시
11	SIGSEGV	코어 덤프	세그먼테이션 폴트 시
12	SIGSYS	코어 덤프	system call 잘못했을때
13	SIGPIPE	코어 덤프	파이프 처리 잘못했을때
14	SIGALRM	코어 덤프	알람에 의해 발생함.
16	SIGUSR1	종료	사용자 정의 시그널1
17	SIGUSR2	종료	사용자 정의 시그널2
18	SIGCHLD	무시	자식 프로세스(child process) 상태 변할때
23	SIGSTOP	중지	이 시그널을 받으면 SIGCONT시그널을 받을때 까지 프로세스 중지.
24	SIGTSTP	중지	ctrl + z 입력시.
25	SIGCONT	무시	중지된 프로세스 실행시
28	SIGVTALRM	종료	가상 타이머 종료시.

Signals

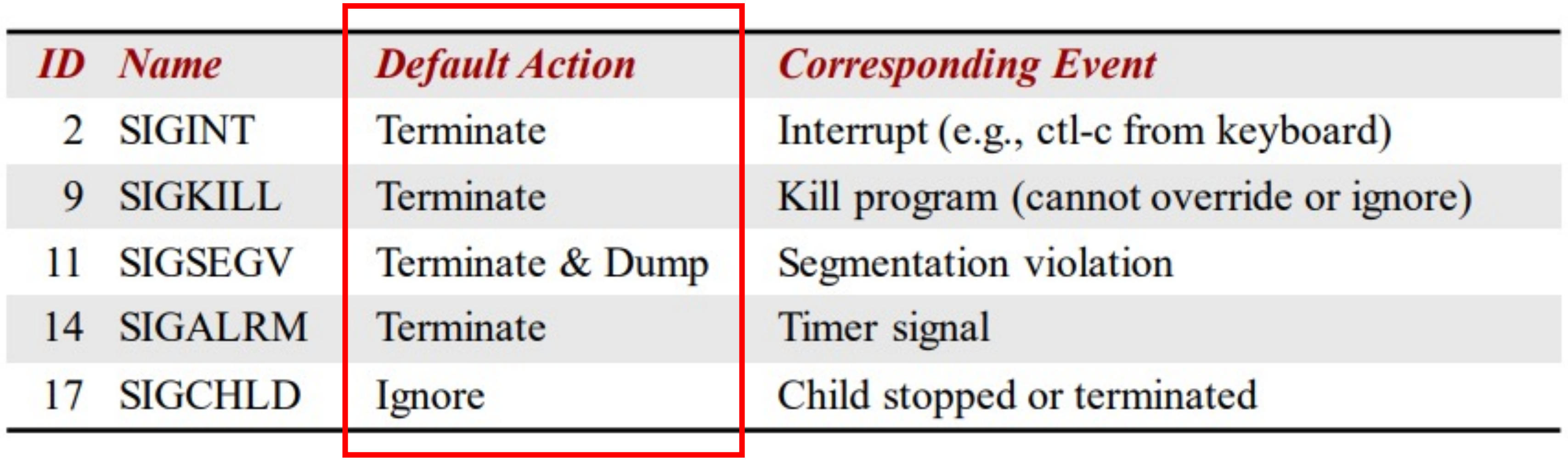
- Each signal ID represents a situation
  - Mostly used signals



ID	*Name*	*Default Action*	*Corresponding Event*
2	SIGINT	Terminate	Interrupt (e.g., ctl-c from keyboard)
9	SIGKILL	Terminate	Kill program (cannot override or ignore)
11	SIGSEGV	Terminate & Dump	Segmentation violation
14	SIGALRM	Terminate	Timer signal
17	SIGCHLD	Ignore	Child stopped or terminated

Signals

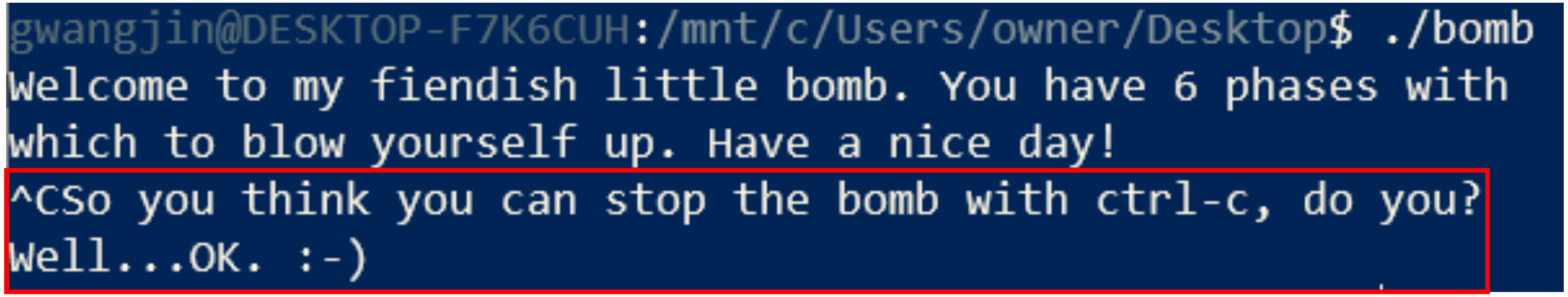
- For each signal, programs do ***default*** action



ID	Name	Default Action	Corresponding Event
2	SIGINT	Terminate	Interrupt (e.g., ctl-c from keyboard)
9	SIGKILL	Terminate	Kill program (cannot override or ignore)
11	SIGSEGV	Terminate & Dump	Segmentation violation
14	SIGALRM	Terminate	Timer signal
17	SIGCHLD	Ignore	Child stopped or terminated

Signals

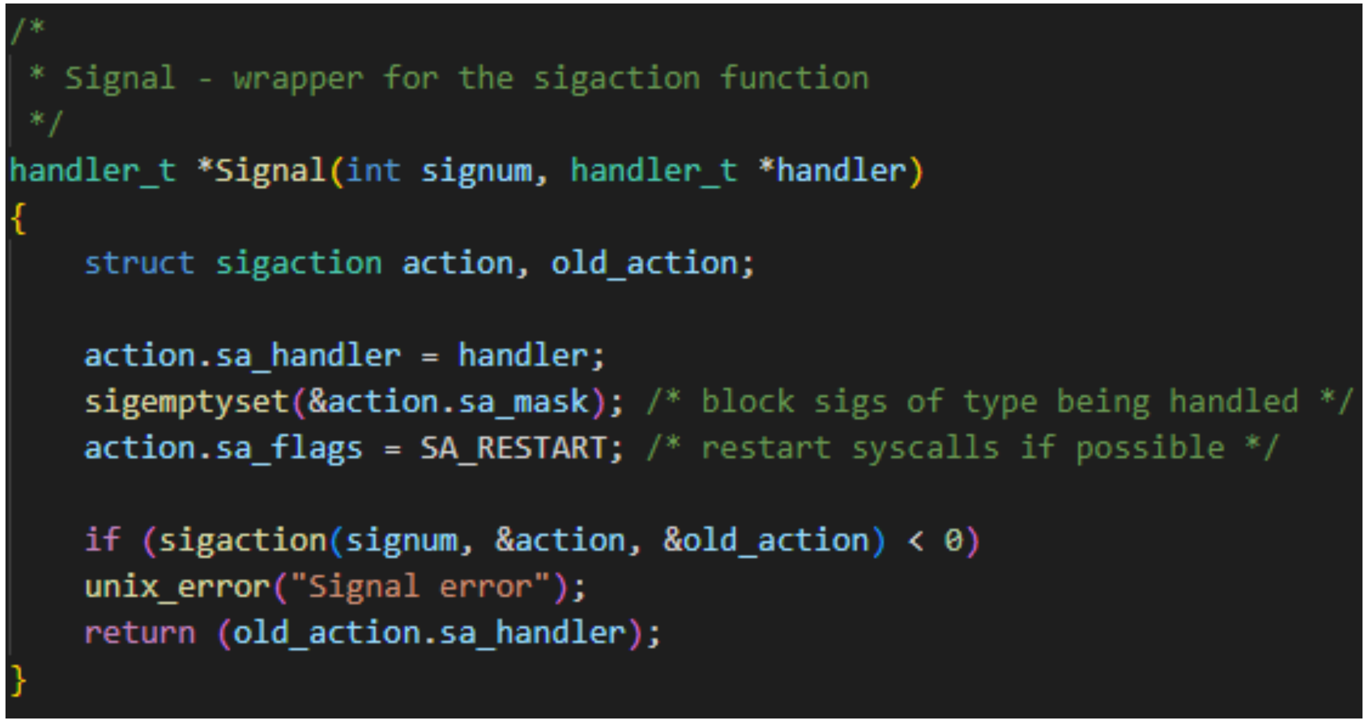
- For each signal, users can define action
  - E.g., Bomblab



```
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
^CSo you think you can stop the bomb with ctrl-c, do you?
Well...OK. :-)
```

Signals

- For each signal, users can define action
  - E.g., Shell Lab



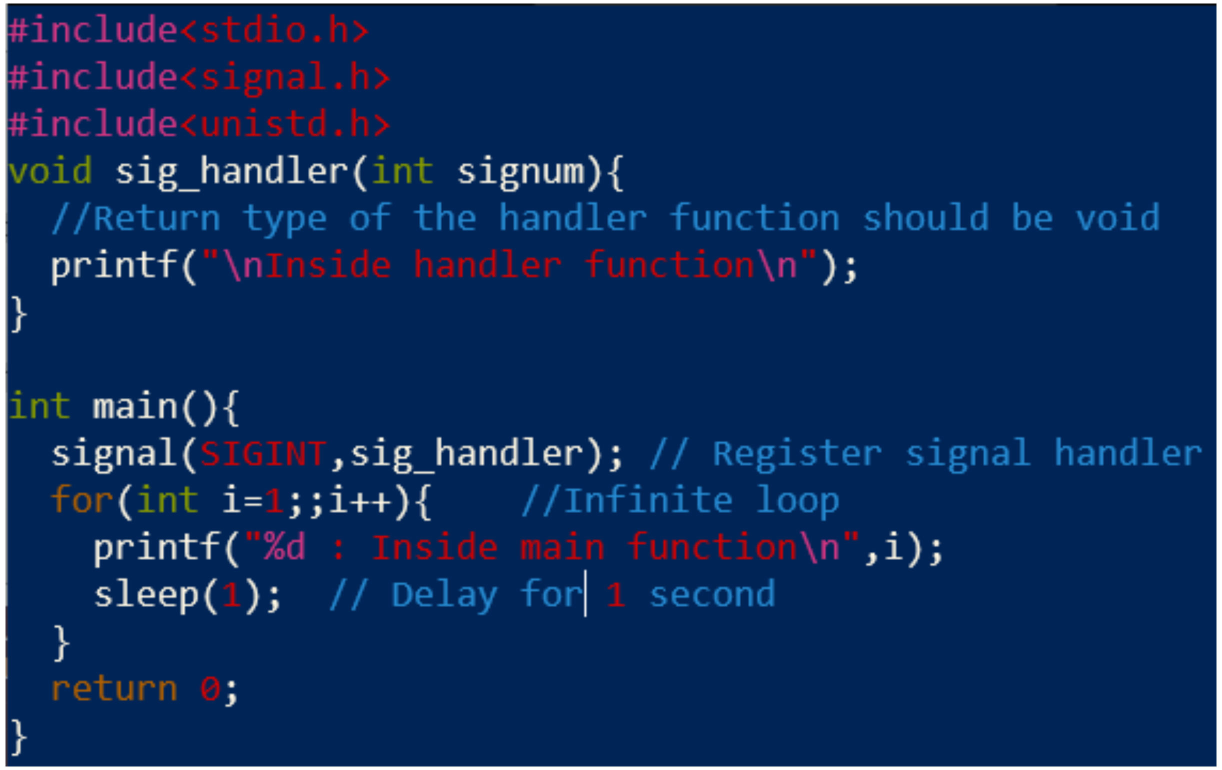
```
/*
 * Signal - wrapper for the sigaction function
 */
handler_t *Signal(int signum, handler_t *handler)
{
    struct sigaction action, old_action;

    action.sa_handler = handler;
    sigemptyset(&action.sa_mask); /* block sigs of type being handled */
    action.sa_flags = SA_RESTART; /* restart syscalls if possible */

    if (sigaction(signum, &action, &old_action) < 0)
    unix_error("Signal error");
    return (old_action.sa_handler);
}
```

Signal Handling

- Simple signal handler
  - **`include <signal.h>`**
  - It replaces handler
  - ***"DO NOT RUN THIS CODE"***

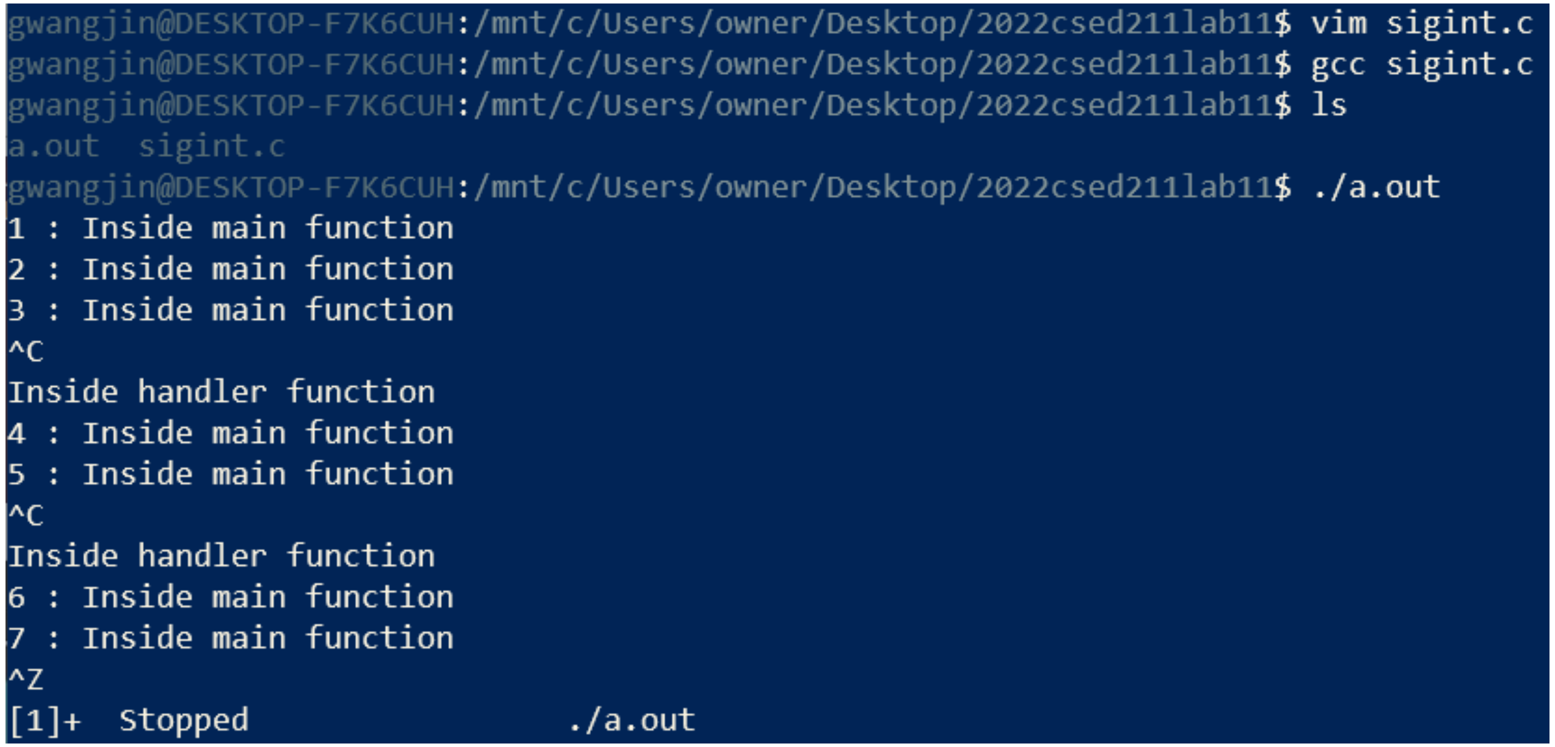


```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
void sig_handler(int signum){
  //Return type of the handler function should be void
  printf("\nInside handler function\n");
}

int main(){
  signal(SIGINT,sig_handler); // Register signal handler
  for(int i=1;;i++){    //Infinite loop
    printf("%d : Inside main function\n",i);
    sleep(1);  // Delay for 1 second
  }
  return 0;
}
```

Signal Handling

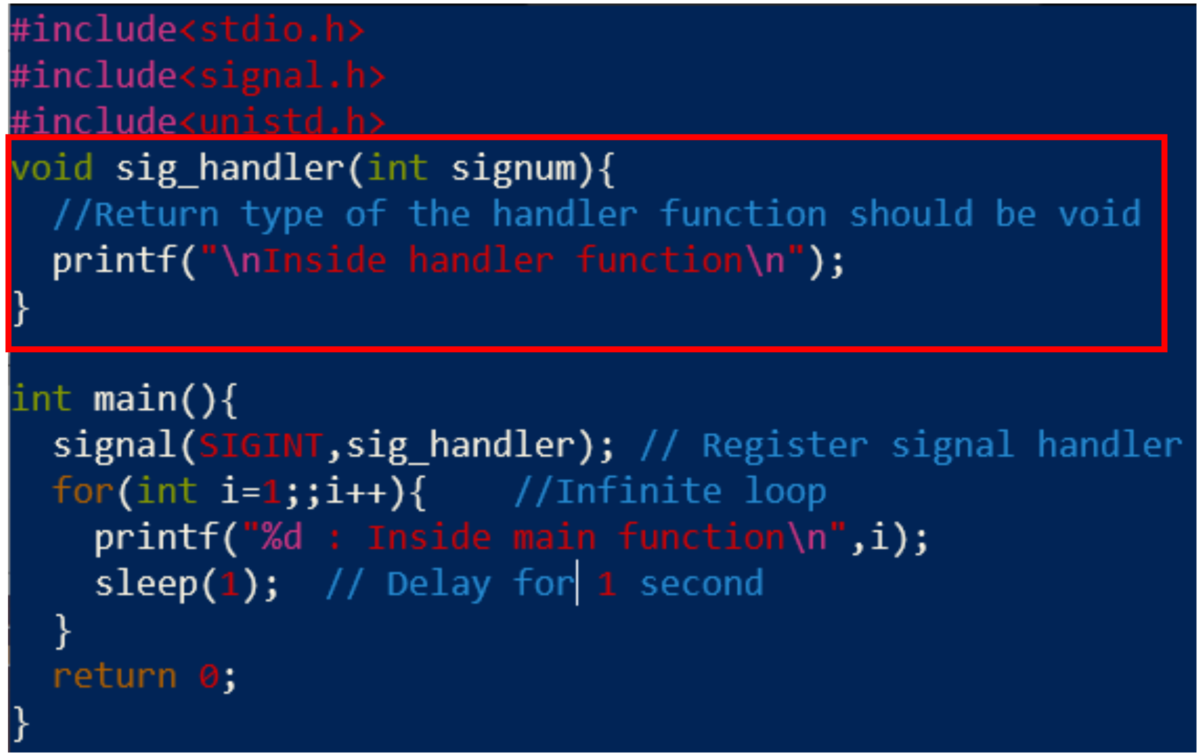
- **`Sig_handler`** only prints
  - We can't <u>kill</u> this program



```
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ vim sigint.c
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ gcc sigint.c
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ ls
a.out  sigint.c
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ ./a.out
1 : Inside main function
2 : Inside main function
3 : Inside main function
^C
Inside handler function
4 : Inside main function
5 : Inside main function
^C
Inside handler function
6 : Inside main function
7 : Inside main function
^Z
[1]+  Stopped                 ./a.out
```

Signal Handling

- In signal.h
  - Main routine that parses and interprets the command line
- **`void (*signal(int signum, void (*handler)(int)))(int)`**
  - **`int signum`**
    - Signal number (e.g., **`SIGINT = 2`**)
  - **`void (*handler) (int)`**
    - Handler function
- Returns : **`void *() (int)`**
  - **`Original handler`**

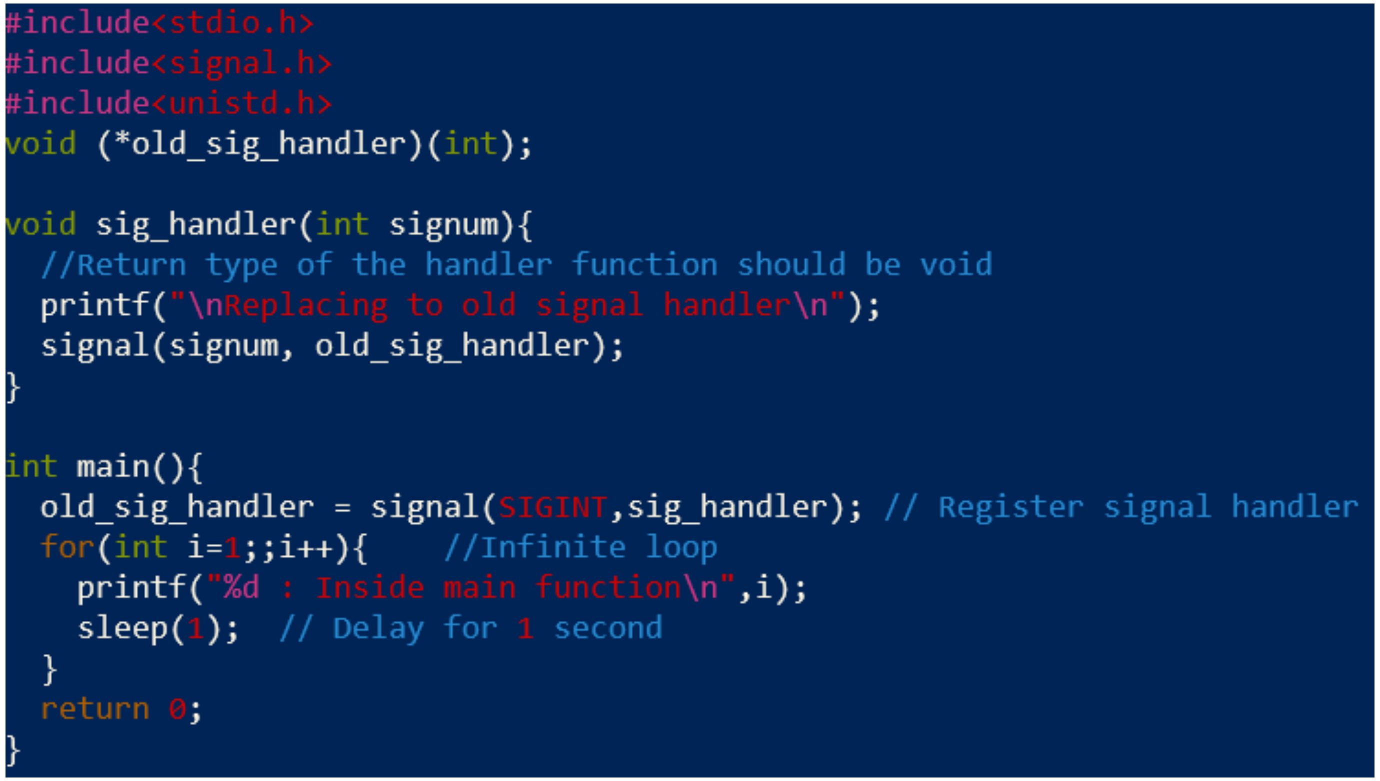


```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
void sig_handler(int signum){
  //Return type of the handler function should be void
  printf("\nInside handler function\n");
}

int main(){
  signal(SIGINT,sig_handler); // Register signal handler
  for(int i=1;;i++){    //Infinite loop
    printf("%d : Inside main function\n",i);
    sleep(1);  // Delay for 1 second
  }
  return 0;
}
```

Signal Handling

- Returns : **`void *() (int)`**
  - **`Original handler`**
- Using return value of signal
  - We can switch handlers



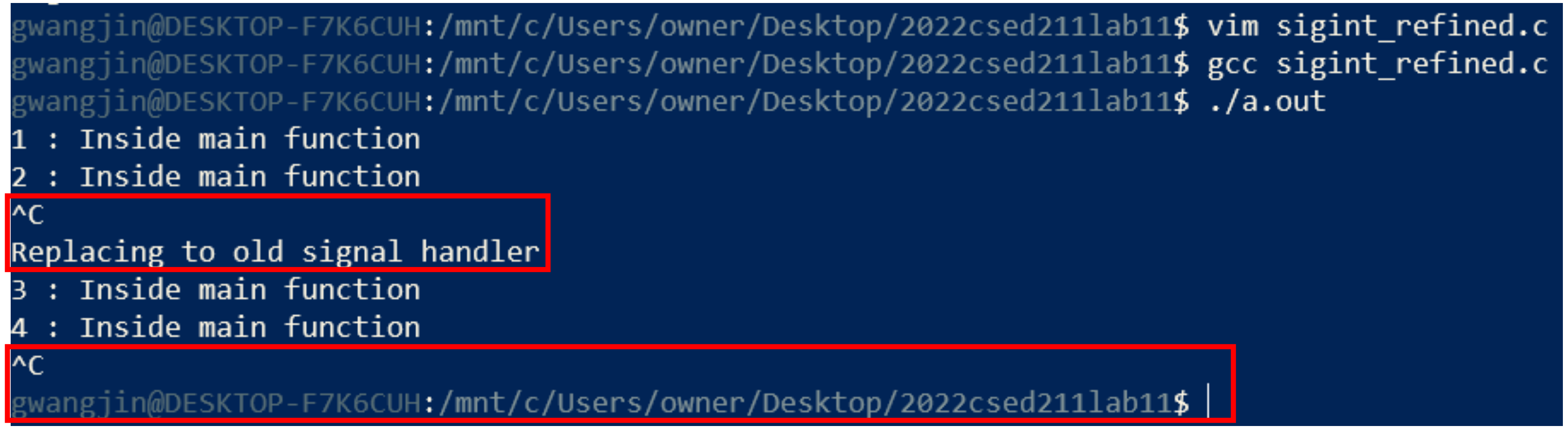
```
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
void (*old_sig_handler)(int);

void sig_handler(int signum){
  //Return type of the handler function should be void
  printf("\nReplacing to old signal handler\n");
  signal(signum, old_sig_handler);
}

int main(){
  old_sig_handler = signal(SIGINT,sig_handler); // Register signal handler
  for(int i=1;;i++){    //Infinite loop
    printf("%d : Inside main function\n",i);
    sleep(1);  // Delay for 1 second
  }
  return 0;
}
```

Signal Handling

- Returns : **`void *() (int)`**
  - **`Original handler`**
- Using return value of signal
  - We can switch handlers



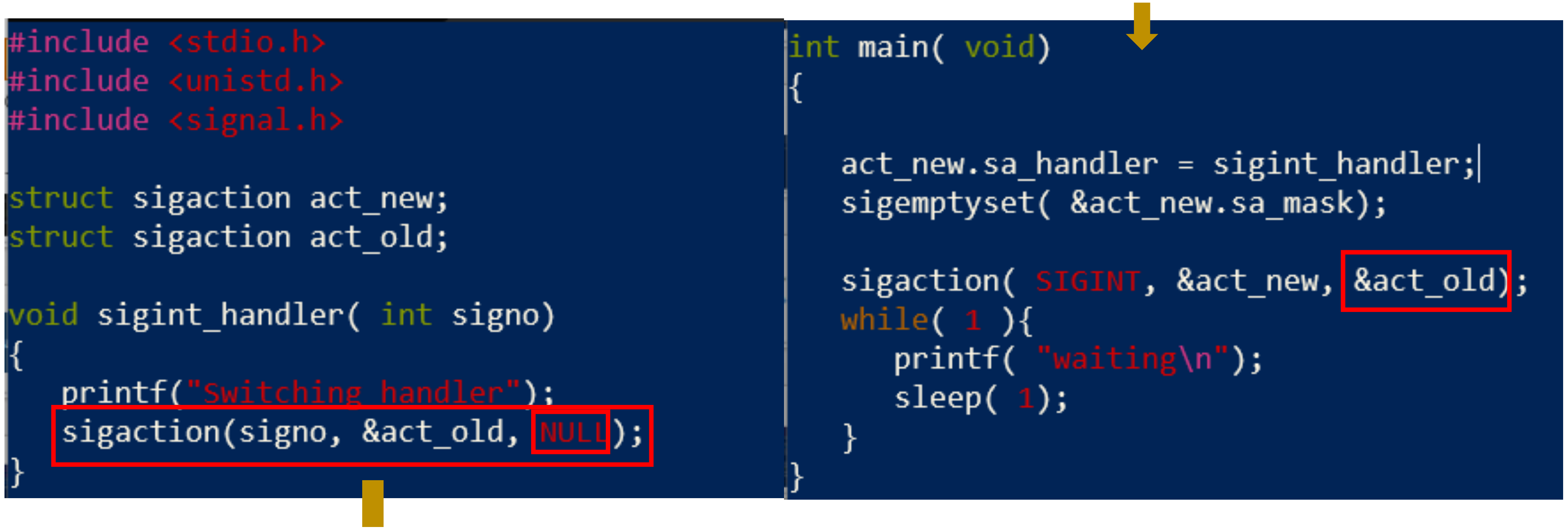
```
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ vim sigint_refined.c
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ gcc sigint_refined.c
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ ./a.out
1 : Inside main function
2 : Inside main function
^C
Replacing to old signal handler
3 : Inside main function
4 : Inside main function
^C
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$
```

Signal Handling

- In **`singal.h`**
  - **`Void (*signal(int signum, void (*handler)(int))) (int)`**
    - **Inefficient**
- In **`singal.h`**
  - **`int sigaction (int signum, const struct sigaction *act, struct sigaction *oldact)`**
    - **`int signum`**
    - **`struct sigaction *act`**
    - **`struct sigaction *oldact`**
  - Returns **`int 0 (success), -1 (fail)`**

Signal Handling

- In this code
  - Original **SIGINT** handler is ***act_old***
  - When new **SIGINT** handler called, replace handler to **act_old**
  - Forget our new **SIGINT** handler



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

struct sigaction act_new;
struct sigaction act_old;

void sigint_handler( int signo)
{
   printf("Switching handler");
   sigaction(signo, &act_old, NULL);
}
```

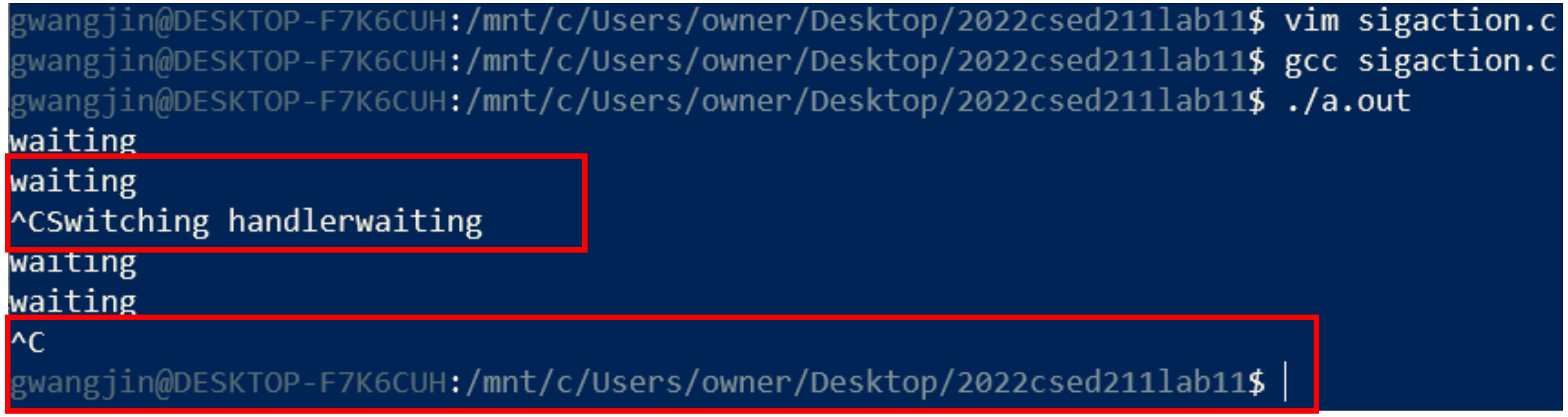
```
int main( void)
{

   act_new.sa_handler = sigint_handler;
   sigemptyset( &act_new.sa_mask);

   sigaction( SIGINT, &act_new, &act_old);
   while( 1 ){
      printf( "waiting\n");
      sleep( 1);
   }
}
```

Signal Handling

- In this code
  - Original **`SIGINT`** handler is ***`act_old`***
  - When new **`SIGINT`** handler called, replace handler to **`act_old`**
  - Forget our new **`SIGINT`** handler



```
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ vim sigaction.c
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ gcc sigaction.c
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ ./a.out
waiting
waiting
^CSwitching handlerwaiting
waiting
waiting
^C
gwangjin@DESKTOP-F7K6CUH:/mnt/c/Users/owner/Desktop/2022csed211lab11$ 
```

QUIZ 1

QUIZ 2