TCP 소켓 프로그래밍 5

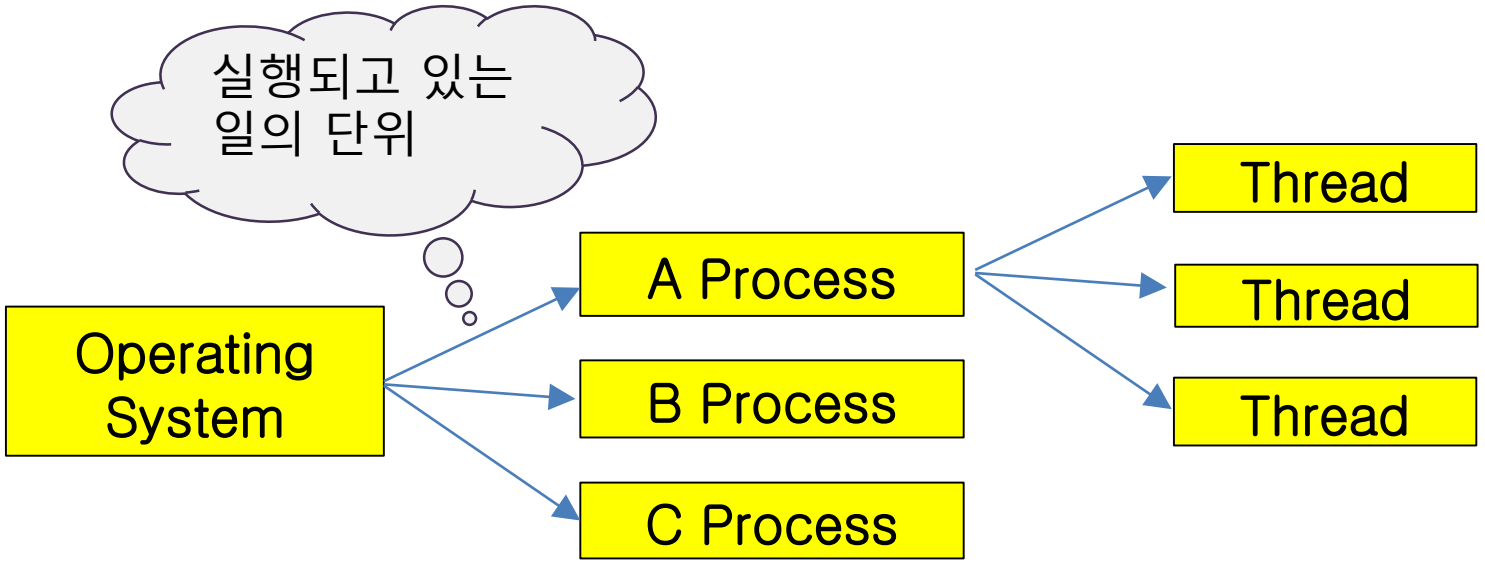
컴퓨터 통신 실습

홍석준

프로세스(Process)와 쓰레드(Thread)

❑ 프로세스와 쓰레드

- 쓰레드는 프로세스의 장점을 지니면서도 어느정도 단점을 극복한 '경량화된 프로세스'
- 프로세스와 쓰레드의 차이점
  - ✓ 쓰레드는 프로세스와 달리 공유되는 메모리 공간을 가지고 있음
  - ✓ 쓰레드 기반 프로그래밍 : 많은 주의를 기울여야 함



< 프로세스와 쓰레드>

프로세스(Process)와 쓰레드(Thread)

1. 경량화 된 프로세스

- ✓ 프로세스와 마찬가지로 동시 실행이 가능
- ✓ 프로세스의 단점을 극복하기 위해 등장

2. 프로세스와의 차이점

- ✓ 스택을 제외한 나머지 메모리 영역을 공유
- ✓ 보다 간단한 컨텍스트 스위칭
- ✓ 일부 메모리를 공유하므로 쓰레드간 통신이 편리

쓰레드 생성하기

POSIX 쓰레드 생성함수

❑ pthread_create 함수

```
#include <pthread.h>

int pthread_create(pthread_t * thread, pthread_attr_t * attr, void *(*start
routine)(void*), void * arg)
```

- thread : 생성된 쓰레더의 ID를 저장할 변수의 포인터를 인자로 전달. 함수가 호출되고 나면 쓰레드가 생성되는데, 생성되는 모든 쓰레드는 프로세스처럼 ID를 할당받게 됨
- attr : 생성하고자 하는 쓰레드의 특성(attribute)을 설정할 때 사용, 일반적을 NULL을 전달
- start_routine : 함수 포인터를 인자로 요구. 리턴 타입이 void *이고 인자도 void*인 함수를 가리키는 포인터, 쓰레드가 생성되고 나서 실행해야하는 함수 루틴을 설정해 줌. 따라서, 쓰레드가 생성되자마자 여기서 인자로 전달된 함수를 호출하게 됨. 이 함수가 종료와 동시에 쓰레드도 소멸
- arg : 쓰레드에 의해 호출되는 함수(start_routine 포인터가 가리키는 함수)에 전달하고자 하는 인자값을 넘겨줌

쓰레드 생성하기

POSIX 쓰레드 생성함수

❑ pthread_create 함수

```
#include <pthread.h>

int pthread_create(pthread_t * thread, pthread_attr_t * attr, void *(*start
routine)(void*), void * arg)
```

– 리턴값 : 성공시 0, 실패 시 이외의 값을 리턴

쓰레드 생성하기

POSIX 쓰레드 생성함수

- gedit thread1.c로 프로그램 작성(그림 참조)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>

void *thread_function(void *arg);

int main(int agrc, char **argv)
{
        int state;
        pthread_t t_id;
        void * t_return;

        state = pthread_create(&t_id, NULL, thread_function, NULL);
        if(state != 0)
        {
                puts("thread create error!!!\n");
                exit(1);
        }
        printf("created thread id : %lu \n", t_id);
        sleep(3);
        puts("main function end");

        return 0;
}
```

(계속)

쓰레드 생성하기

POSIX 쓰레드 생성함수

- gedit thread1.c로 프로그램 작성(그림 참조)

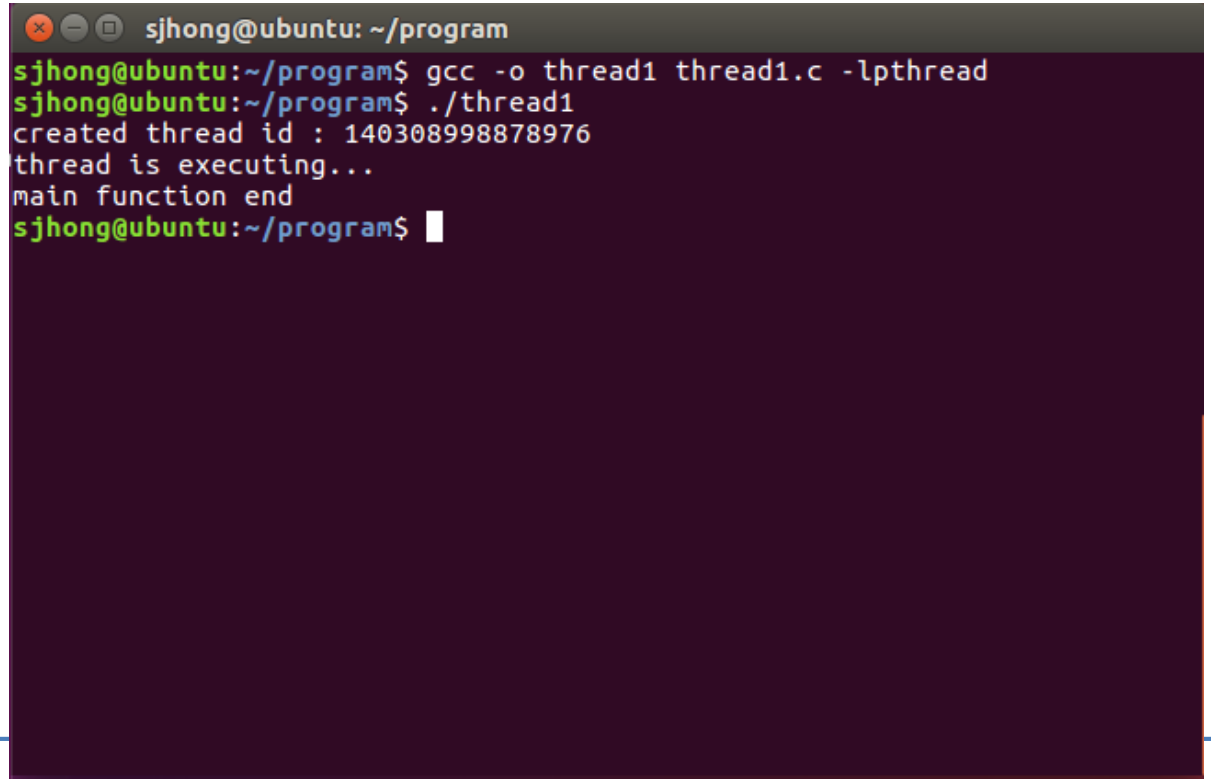
```
void *thread_function(void *arg)
{
        int i;
        for(i=0;i<3;i++){
                sleep(2);
                puts("thread is executing....");
        }
}
```

(끝)

쓰레드 생성하기

POSIX 쓰레드 생성함수

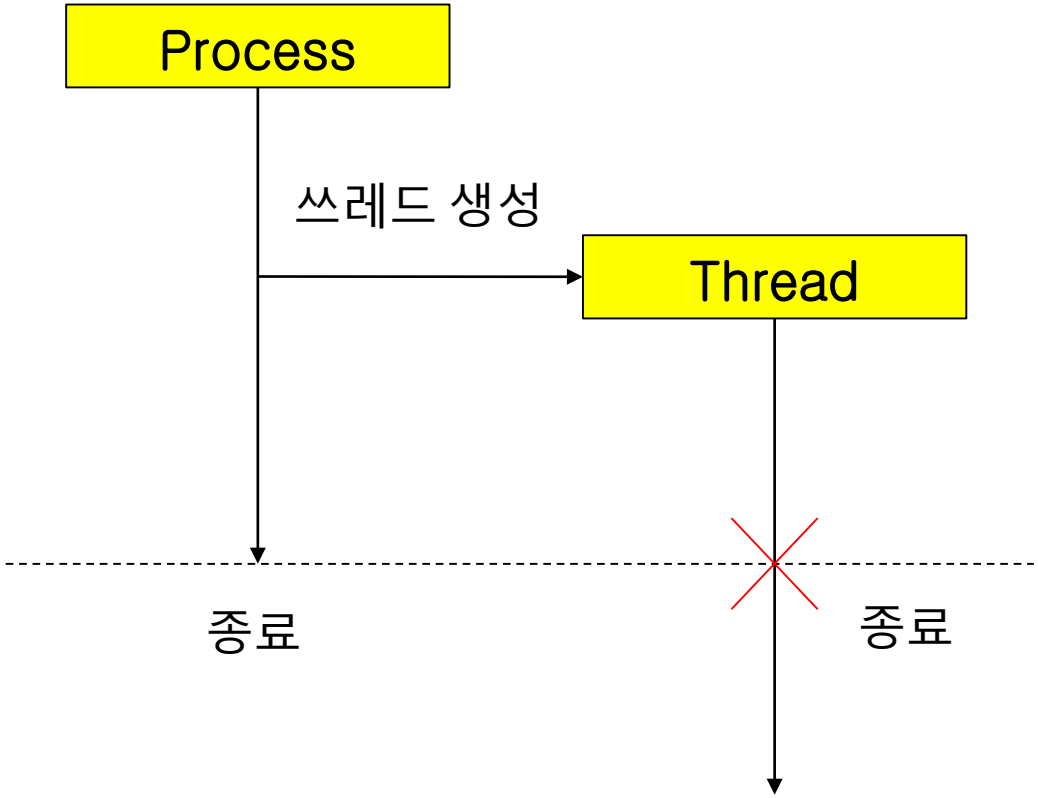
- **gcc –o thread1 thread1.c -lpthread로 컴파일 및 실행**
  - 쓰레드와 관련된 라이브러리가 기본적으로 링크되어 있지 않음으로 “-lpthread” 라이브러리 옵션으로 쓰레드 라이브러리를 링크해야함.
  - 실행 결과 : “thread is executing”이 한번만 실행되고 끝남



쓰레드 생성하기

프로세스와 쓰레드의 종료 관계

❑ **thread1.c프로그램의 흐름**



쓰레드 생성하기

POSIX 쓰레드 join함수

❑ pthread_join 함수

```
#include <pthread.h>

int pthread_join(pthread_t th, void **thread_return);
```

- th : th에 인자로 들어오는 ID의 쓰레드가 종료할 때까지 실행을 지연시킴
- thread_return : 쓰레드가 종료 시 반환하는 값에 접근할 수 있는 2차원 포인터

쓰레드 생성하기

POSIX 쓰레드 join함수

❑ gedit thread2.c로 프로그램 작성(그림 참조)

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>

void *thread_function(void *arg);

int main(void)
{
        int state;
        pthread_t t_id;
        void * t_return;

        state = pthread_create(&t_id, NULL, thread_function, NULL);
        if(state != 0)
        {
                puts("thread create error!!!");
                exit(1);
        }

        printf("created thread id : %lu \n", t_id);
        state = pthread_join(t_id, &t_return);
        if(state != 0)
        {
                puts("thread join error!!!");
                exit(1);
        }

        sleep(3);
        printf("main function end. thread return %s", (char *)t_return);
        free(t_return);
        return 0;

}
```

(계속)

쓰레드 생성하기

POSIX 쓰레드 join함수

❑ **gedit thread2.c로 프로그램 작성(그림 참조)**

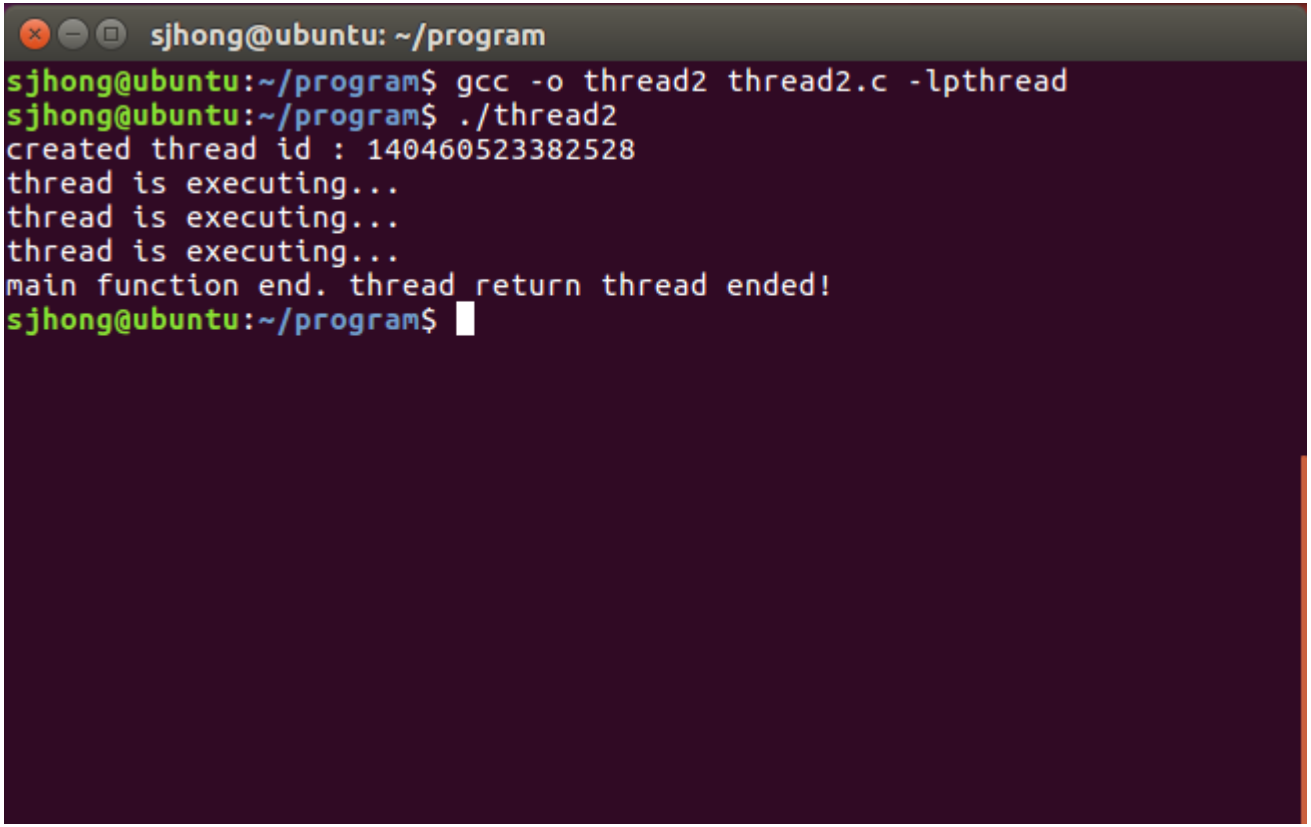
```
void *thread_function(void *arg)
{
        int i;
        char * p = (char *) malloc(20*sizeof(char));
        strcpy(p, "thread end!\n");

        for(i=0;i<3;i++){
                sleep(2);
                puts("thread is executing....");
        }

        return p;
}
```

(끝)

쓰레드 생성하기

POSIX 쓰레드 join함수

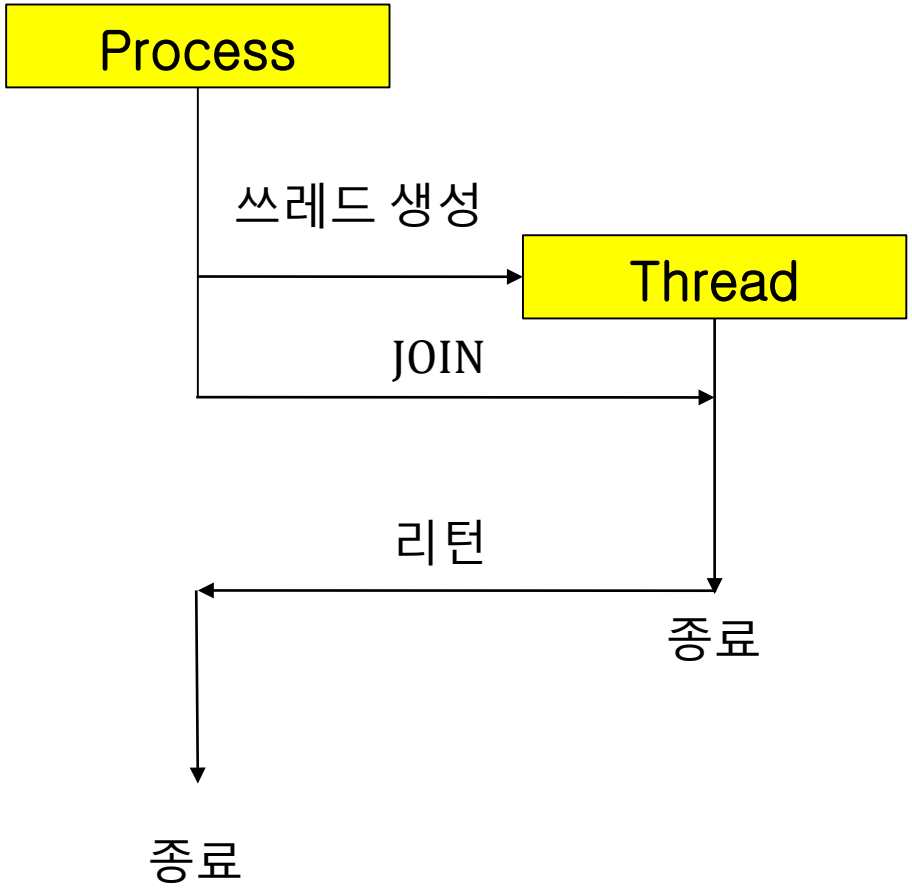
- **gcc –o thread2 thread2.c -lpthread로 컴파일 및 실행**
  - "thread is executing"이 정상적으로 세 번 다 출력하고 종료됨.



쓰레드 생성하기

프로세스와 쓰레드의 JOIN

❑ **thread2.c프로그램의 흐름**



쓰레드 생성하기

동시에 실행하는 멀티쓰레드

❑ 다중 쓰레드 생성 모델

- 1부터 10까지 덧셈하는 프로그램
- 두 개의 쓰레드 생성, 하나는 1부터 5까지 덧셈, 다른 하나는 6부터 10까지 덧셈하여 그 결과를 main에서 참조

쓰레드 생성하기

동시에 실행하는 멀티쓰레드

❑ 다중 쓰레드 생성 모델 예제(thread3.c)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>

void *thread_summation(void *arg);

int sum=0;
int sum1[]={1,5};
int sum2[]={6,10};

int main(int argc, char **argv)
{
        pthread_t id_t1, id_t2;
        void * t_return;

        pthread_create(&id_t1, NULL, thread_summation, (void *)sum1);
        pthread_create(&id_t2, NULL, thread_summation, (void *)sum2);

        pthread_join(id_t1, &t_return);
        pthread_join(id_t2, &t_return);

        printf("main function end, sum = %d \n", sum);

        return 0;
}
```

(계속)

쓰레드 생성하기

동시에 실행하는 멀티쓰레드

❑ 다중 쓰레드 생성 모델 예제(thread3.c)

```
void *thread_summation(void *arg)
{
        int start = ((int*)arg)[0];
        int end = ((int*)arg)[1];

        printf("thread start - start : %d, end : %d\n", start, end);

        for(;start<=end;start++)
        {
                sum+=start;
        }
}
```

(끝)

쓰레드 생성하기

동시에 실행하는 멀티쓰레드

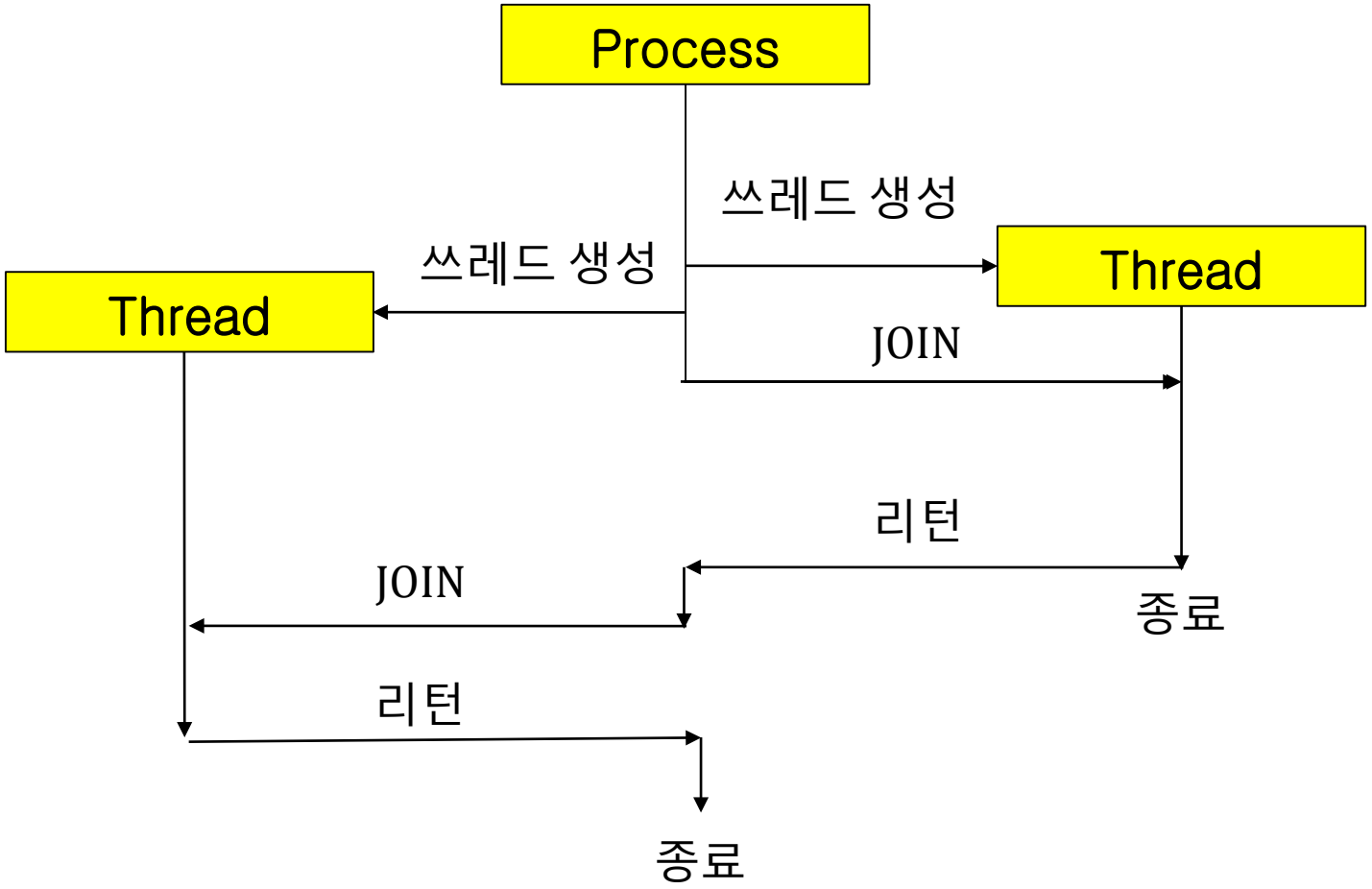
- **gcc –o thread3 thread3.c -lpthread로 컴파일 및 실행**
  - 1부터 10까지 더해서 최종 결과 sum은 55가 됨

```
sjhong@ubuntu:~/program$ gcc -o thread3 thread3.c -lpthread
sjhong@ubuntu:~/program$ ./thread3
thread start - start : 6, end : 10
thread start - start : 1, end : 5
main function end, sum = 55
sjhong@ubuntu:~/program$
```

쓰레드 생성하기

동시에 실행하는 멀티쓰레드

❑ 다중 쓰레드 생성 모델



쓰레드 기반 TCP 소켓 프로그램 컴파일 및 실행

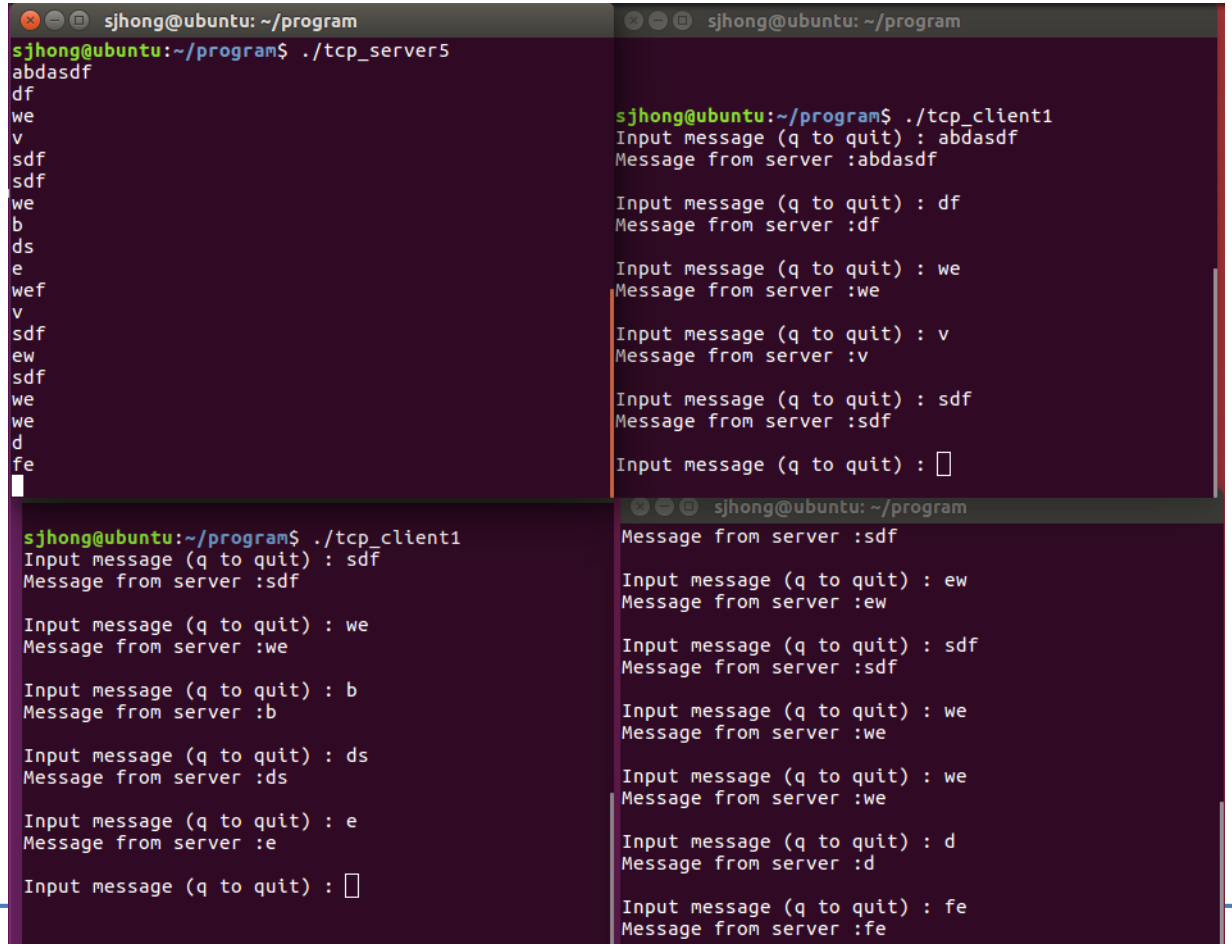
TCP 소켓 프로그램 컴파일 및 실행

- **tcp_server5.c파일로 작성하기**
  - Hint 기존에 작성한 tcp_server2.c와 thread3.c 코드를 이용
- **gcc –o tcp_server5 tcp_server5.c -lphtread로 컴파일 및 실행 확인**

쓰레드 기반 TCP 소켓 프로그램 컴파일 및 실행

TCP 소켓 프로그램 컴파일 및 실행

- ❑ 아래 그림처럼 실행되는지 확인(클라이언트는 기존 tcp_client1사용)
  - 하나의 서버 실행 후 여러 클라이언트 접속 및 통신 가능한지를 확인




Thank you for your attention !!