Thr1.c

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
void * func(void *i);
int main(void)
{
	pthread_t thr;
	printf("\n");
	pthread_create(&thr, NULL, func, NULL);
	printf("Main thread ...\n");
	pthread_join(thr, NULL);
	printf("Joined ...\n");
	}
void * func(void *arg)
	{
	printf("Child thread ...\n");
	sleep(60);
	pthread_exit(NULL);
}
```

```
(base) 202020827@cslinux:~/os/thread$ gcc -o thr1 thr1.c -lpthread
(base) 202020827@cslinux:~/os/thread$ ./thr1 &
[1] 1309127

Main thread ...
Child thread ...
(base) 202020827@cslinux:~/os/thread$ ps
    PID TTY          TIME CMD
1302152 pts/69   00:00:00 bash
1309127 pts/69   00:00:00 thr1
1309147 pts/69   00:00:00 ps
(base) 202020827@cslinux:~/os/thread$ ps -L
    PID     LWP TTY          TIME CMD
1302152 1302152 pts/69   00:00:00 bash
1309127 1309127 pts/69   00:00:00 thr1
1309127 1309129 pts/69   00:00:00 thr1
1309157 1309157 pts/69   00:00:00 ps
(base) 202020827@cslinux:~/os/thread$ Joined ...
```

1. **프로세스 개수**
   → thr1 실행으로 1 개 프로세스 생성
2. **스레드 개수**
   → 2 개 스레드 (main + child)

LWP:

- 커널이 관리하는 스레드 ID
- 여러 LWP 가 하나의 PID 를 공유 (스레드는 PID 하나에 묶임)

3. 각 스레드 역할

- 자식 스레드: 메시지 출력 → 60 초 대기 → 종료
- 메인 스레드: 메시지 출력 → pthread_join()로 자식 대기 → 종료 메시지 출력

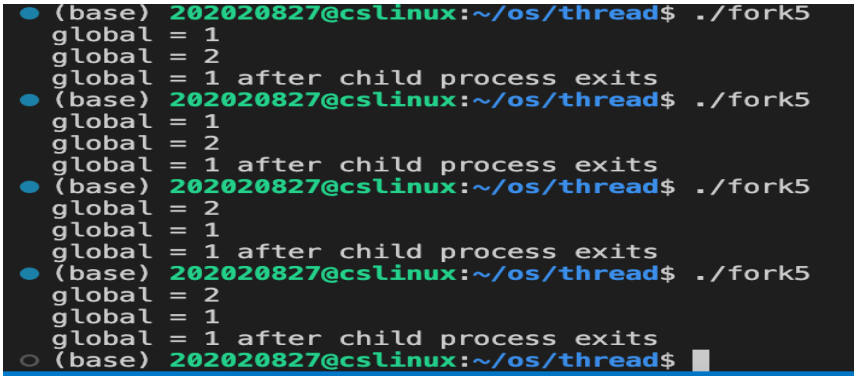
4. 함수 역할

- pthread_join(): 자식 스레드 종료 대기 및 동기화
- pthread_exit(): 스레드 명시적 종료 및 상태 전달

Fork5.c

```c
#include <unistd.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>

void X(int i);
int global = 0;
int main(void)
   {
   int status;
   if (fork() != 0) {
   X(1);
   waitpid(-1, &status, 0);
   } else {
   X(2);
   exit(0);
   }
   printf("global = %d after child process exits\n", global);
   }
void X(int i)
   {
   global = i;
   printf("global = %d\n", global);
}
```



실행 결과는?

1. 부모 프로세스가 먼저 실행되어 global = 1, 자식 프로세스가 나중에 실행되어 global = 2가 출력된다.
2. 부모는 waitpid()로 자식이 끝나길 기다린 후, global = 1 after child process exits를 출력한다.
3. fork() 이후 두 프로세스가 병렬로 실행되기 때문에, global = 1과 global = 2의 출력 순서는 상황에 따라 바뀔 수 있다.

부모와 자식 프로세스는 변수 global 값을 공유하는가?

공유하지 않음
→ fork()는 메모리를 복사하기 때문에, 부모와 자식은 서로 다른 global 변수를 가짐

Thr2.c

```c
#include <pthread.h>
#include <stdio.h>
void * X(void *p);
void Y(int j);
int global = 0;
int main()
{
pthread_t t1;
    void *status;
    printf("\n");
    pthread_create(&t1, NULL, X, NULL);
    Y(1);
    pthread_join(t1, &status);
    printf("global=%d in main thread after the child thread exiting\n",
    global);
    }

void * X(void *p)
    {
    int i;
    do {
```

```
    global = 2;
    for (i = 0; i < 100000; i++) ;
    printf("Child thread: global=%d\n", global);
    } while (1);
    pthread_exit((void *)NULL);
    }

void Y(int j)
{
    int i;
    do {
    global = j;
    for (i = 0; i < 100000; i++) ;
    printf("Main thread: global=%d\n", global);
    } while (1);
}
```

```
Child thread: global=2
Main thread: global=2
Child thread: global=2
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=2
Main thread: global=2
Child thread: global=2
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=1
Main thread: global=2
Child thread: global=2
Main thread: global=2
```

1. 실행 결과

global = 1, global = 2 출력 순서는 랜덤 (부모/자식 순서에 따라 다름)

> 마지막 출력은 항상 global = 1 after child process exits

2. global 변수 공유 여부

- 공유하지 않음
- fork()는 메모리를 복사하므로, 부모/자식은 독립된 global 변수를 가짐
- 자식에서 바꾼 값은 부모에 영향 없음