race-thread.c

```c
#define ITER (50000000)
#include <pthread.h>
#include <stdio.h>
int gvar = 0;

void * X(void *i);
        int main(void)
        {
        pthread_t t1;
        pthread_attr_t attr;
        void *status;
        int j;

        pthread_create(&t1, &attr, X, NULL);
        // Main thread
        ////////////////////////
        for (j = 0; j < ITER; j++) {
        gvar = gvar + 1;
        }
        ////////////////////////

        pthread_join(t1, &status);
        printf("%d\n", gvar);
}

void * X(void *i)
        {
        int j;

        // Child thread
        ////////////////////////
        for (j = 0; j < ITER; j++) {
        gvar = gvar + 1;
        }
        ////////////////////////
        pthread_exit(NULL);
        }
```

```
(base) 202020827@cslinux:~$ ls
os  public_html
(base) 202020827@cslinux:~$ cd os
(base) 202020827@cslinux:~/os$ ls
block  fork  fork-execv  lowlevel_read  process  race-thread  thread
(base) 202020827@cslinux:~/os$ cd race-thread/
(base) 202020827@cslinux:~/os/race-thread$ ./race-thread &
[1] 1566192
(base) 202020827@cslinux:~/os/race-thread$ 51238384
```

- 실행되는 스레드는 총 2 개
    - 메인 스레드와 생성된 자식 스레드
- 각 스레드는 각각 50,000,000 번씩 "gvar = gvar + 1;"을 수행
    - 각 스레드가 gvar 값을 50,000,000 만큼 증가시킴
- 총 증가 합은 100,000,000 이어야 함
    - 하지만 실제 출력값은 100,000,000 과 일치하지 않음
    - 이유: 데이터 레이스 문제로 인해 두 스레드가 동시에 gvar 값을 읽고 갱신하면서 일부 증가가 누락됨
    - 동기화를 적용하지 않아 경쟁 조건이 발생함