운영체제 보고서

빅데이터 학과

20175119 김영식

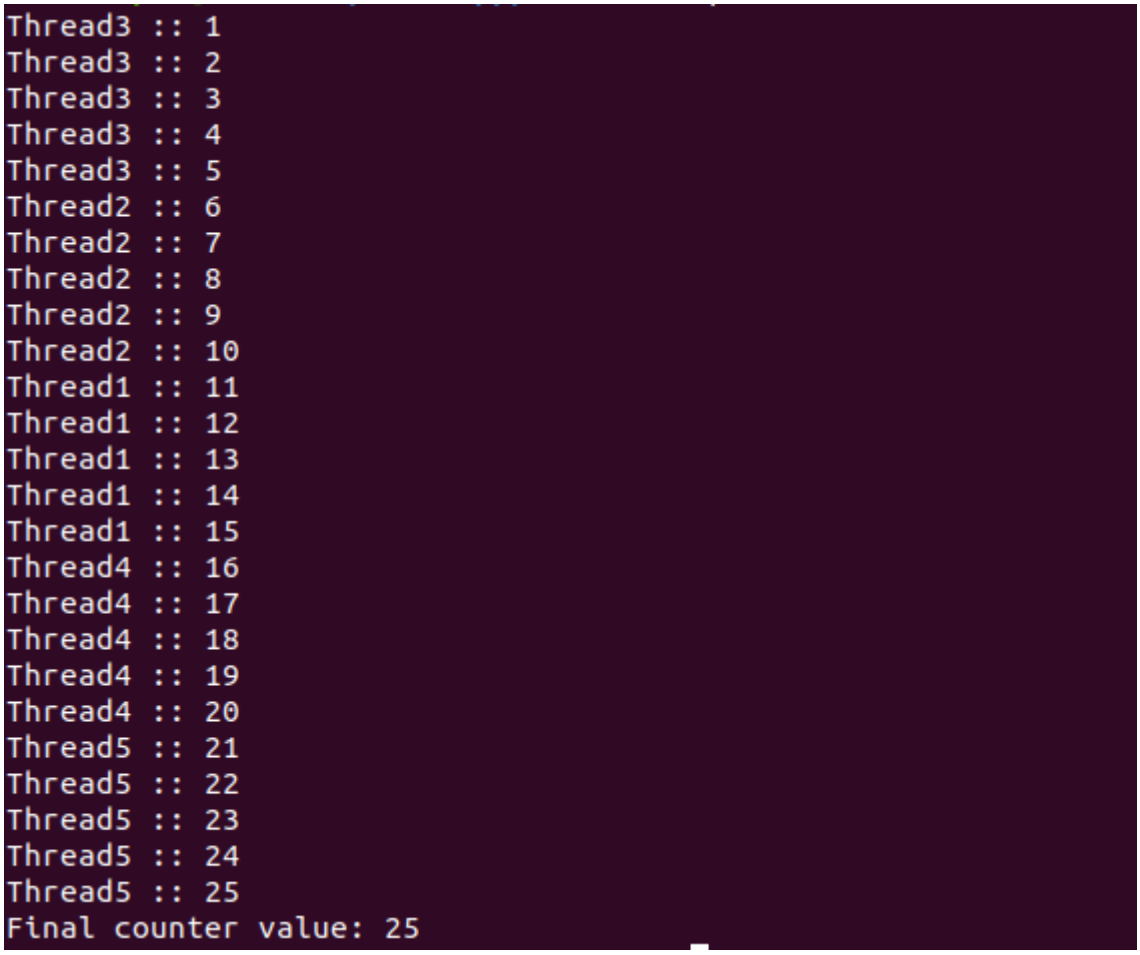
프로그램 코드

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

#define NTHREADS 5

pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;

int counter = 0;

void *fun(int data)
{
    for (int i = 0; i < 100000; i++)
    {
        pthread_mutex_lock(&mutex1);

        counter++;
        printf("Thread%d :: %d\n", data,counter);

        pthread_mutex_unlock(&mutex1);
    }
}

int main()

{

    pthread_t thread_id[NTHREADS];

    for (int i = 0; i < NTHREADS; i++)

    {
        pthread_create(&thread_id[i], NULL, fun, (i+1));
    }

    for (int j = 0; j < NTHREADS; j++)

    {

        pthread_join(thread_id[j], NULL);
    }
    pthread_mutex_destroy(&mutex1);

    printf("Final counter value: %d\n", counter);
}
```

프로그램 설명

총 5개의 쓰레드가 동시에 돌아가도록 코딩했습니다. 뮤텍스를 사용하여 전역변수 counter에는 각각 한 쓰레드 밖에 접근하지 못합니다.

간단하게 쓰레드당 5번씩 돌아보게 코딩하고 pthread_create의 마지막 매개변수를 사용하여 함수에 몇번째 쓰레드가 번호를 출력하는지 확인합니다.

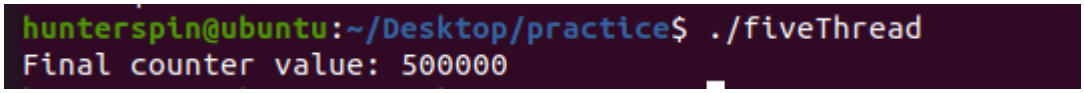


랜덤하게 각각 5번씩 총 25번 값을 출력하는 것을 알 수 있었습니다.

그 후 총 출력 값을 높이고 다시 출력해보니 50만으로 값이 제대로 나오는 것을 확인할 수 있었습니다.

```
Thread4 :: 499953
Thread4 :: 499954
Thread4 :: 499955
Thread4 :: 499956
Thread4 :: 499957
Thread4 :: 499958
Thread4 :: 499959
Thread4 :: 499960
Thread4 :: 499961
Thread4 :: 499962
Thread4 :: 499963
Thread4 :: 499964
Thread4 :: 499965
Thread4 :: 499966
Thread4 :: 499967
Thread4 :: 499968
Thread4 :: 499969
Thread4 :: 499970
Thread4 :: 499971
Thread4 :: 499972
Thread4 :: 499973
Thread4 :: 499974
Thread4 :: 499975
Thread4 :: 499976
Thread4 :: 499977
Thread4 :: 499978
Thread4 :: 499979
Thread4 :: 499980
Thread4 :: 499981
Thread4 :: 499982
Thread4 :: 499983
Thread4 :: 499984
Thread4 :: 499985
Thread4 :: 499986
Thread4 :: 499987
Thread4 :: 499988
Thread4 :: 499989
Thread4 :: 499990
Thread4 :: 499991
Thread4 :: 499992
Thread4 :: 499993
Thread4 :: 499994
Thread4 :: 499995
Thread4 :: 499996
Thread4 :: 499997
Thread4 :: 499998
Thread4 :: 499999
Thread4 :: 500000
Final counter value: 500000
hunterspin@ubuntu:~/Desktop/practice$
```

마지막으로 값만 간단하게 출력하는 것으로 바꾸니



제대로 출력되는 것을 확인 했습니다.

코드 설명

```
#define NTHREADS 5
```

쓰레드 배열을 만들 때 배열의 크기로 사용합니다.

```
pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;
```

뮤텍스 변수를 선언하고 초기화 해줍니다.

```
int counter = 0;
```

쓰레드가 돌아갈 때마다 1씩 증가시켜 총 돌아간 횟수를 저장합니다.

```
void *fun(int data){
	for (int i = 0; i < 100000; i++)
	{
		pthread_mutex_lock(&mutex1);
		counter++;
		printf("Thread%d :: %d₩n", data,counter);
		pthread_mutex_unlock(&mutex1);
	}
}
```

Pthread_create에서 매개변수로 data값을 받아와서 몇번째 쓰레드인지를 출력하고, 값 또한 출력합니다.

Mutex의 lock과 unlock을 이용하여 그 사이에 있는 코드들을 임계지역으로 만들어 줍니다.

```
int main(){
	pthread_t thread_id[NTHREADS]

	for (int i = 0; i < NTHREADS; i++) {
		pthread_create(&thread_id[i], NULL, fun, (i+1));
	}
	for (int j = 0; j < NTHREADS; j++) {
		pthread_join(thread_id[j], NULL);
	}
	pthread_mutex_destroy(&mutex1);
	printf("Final counter value: %d₩n", counter);
}
```

메인문에서는 객체 배열을 생성하고(크기5), for문을 사용하여 쓰레드를 5개 생성하고 매개변수로는 i+1을 넘겨줍니다(1~5 까지의 쓰레드 생성)

join함수는 매개변수로 받은 특정 스레드가 종료하기를 기다렸다가, 종료된 이후 다음을 진행하는 함수이므로 똑같이 쓰레드 만큼 for문을 실행해줍니다.

마지막으로 pthread_mutex_destroy함수를 사용하여 뮤텍스와 관련된 리소스들을 소멸시킵니다.