운영체제 보고서

빅데이터 학과

20175119 김영식

프로그램 코드

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

int cnt;

int flag[2] = {0, 0};

int turn = 0;

void *func0()
{

    for (int i = 0; i < 100000; i++)
    {

        flag[0] = 1;
        turn = 1;
        while (flag[1] == 1 && turn == 1){}

        cnt++;

        printf("cnt1 :: %d\n", cnt);

        flag[0] = 0;
    }
}

void *func1()
{
    for (int i = 0; i < 100000; i++)
    {
        flag[1] = 1;
        turn = 0;
        while (flag[0] == 1 && turn == 0){}

        cnt++;

        printf("cnt2 :: %d\n", cnt);

        flag[1] = 0;
    }
}
```

```c
int main()

{

    pthread_t thread1, thread2;

    int iret1, iret2;

    iret1 = pthread_create(&thread1, NULL, func0, NULL);

    iret2 = pthread_create(&thread2, NULL, func1, NULL);

    pthread_join(thread1, NULL);

    pthread_join(thread2, NULL);

    printf("Thread error: %d\n", 200000 - cnt);

    exit(0);
}
```

코드 분석

int cnt;

int flag[2] = {0, 0};

int turn = 0;

전역 변수 선언하는 코드

Cnt는 쓰레드가 돌아갈때마다 1씩 증가하는 변수이다.

Flag[2]는 두개의 쓰레드가 서로 자신이 기다리고 있는지 여부를 표시한다.

Turn은 0인지 1인지에 따라 두개의 쓰레드가 자신의 턴을 알 수 있게하는 변수이다.

```
void *func0(){
    for (int i = 0; i < 100000; i++){
        flag[0] = 1;
        turn = 1;
        while (flag[1] == 1 && turn == 1){}

        cnt++;

        printf("cnt1 :: %d₩n", cnt);

        flag[0] = 0;
    }
}
```

이 함수는 for문을 10만번 반복하여 cnt 값을 증가시키는데 자신이 호출되면 flag[0]와 turn을 1로 초기화 시킨다.

while (flag[1] == 1 && turn == 1){}

이 부분은 만약 상대방의 턴일 경우 양보하여 자신이 대기하고 있겠다는 코드다.

그후 값을 증가시키고 출력한 뒤 자신의 차례(flag[0])를 0으로 하여 자신의 차례가 끝남을 알린다. → 상대방이 while문에서 벗어나 자신의 차례를 진행한다.

```
void *func1(){
    for (int i = 0; i < 100000; i++){
        flag[1] = 1;
        turn = 0;
        while (flag[0] == 1 && turn == 0){}

        cnt++;

        printf("cnt2 :: %d₩n", cnt);

        flag[1] = 0;
    }
}
```

이 함수도 마찬가지로 for문을 10만번 반복하여 cnt 값을 증가시키는데 자신이 호출되면 flag[1]와 turn을 1로 초기화 시킨다.

while (flag[0] == 1 && turn == 0){}

이 부분은 만약 상대방의 턴일 경우 양보하여 자신이 대기하고 있겠다는 코드다.

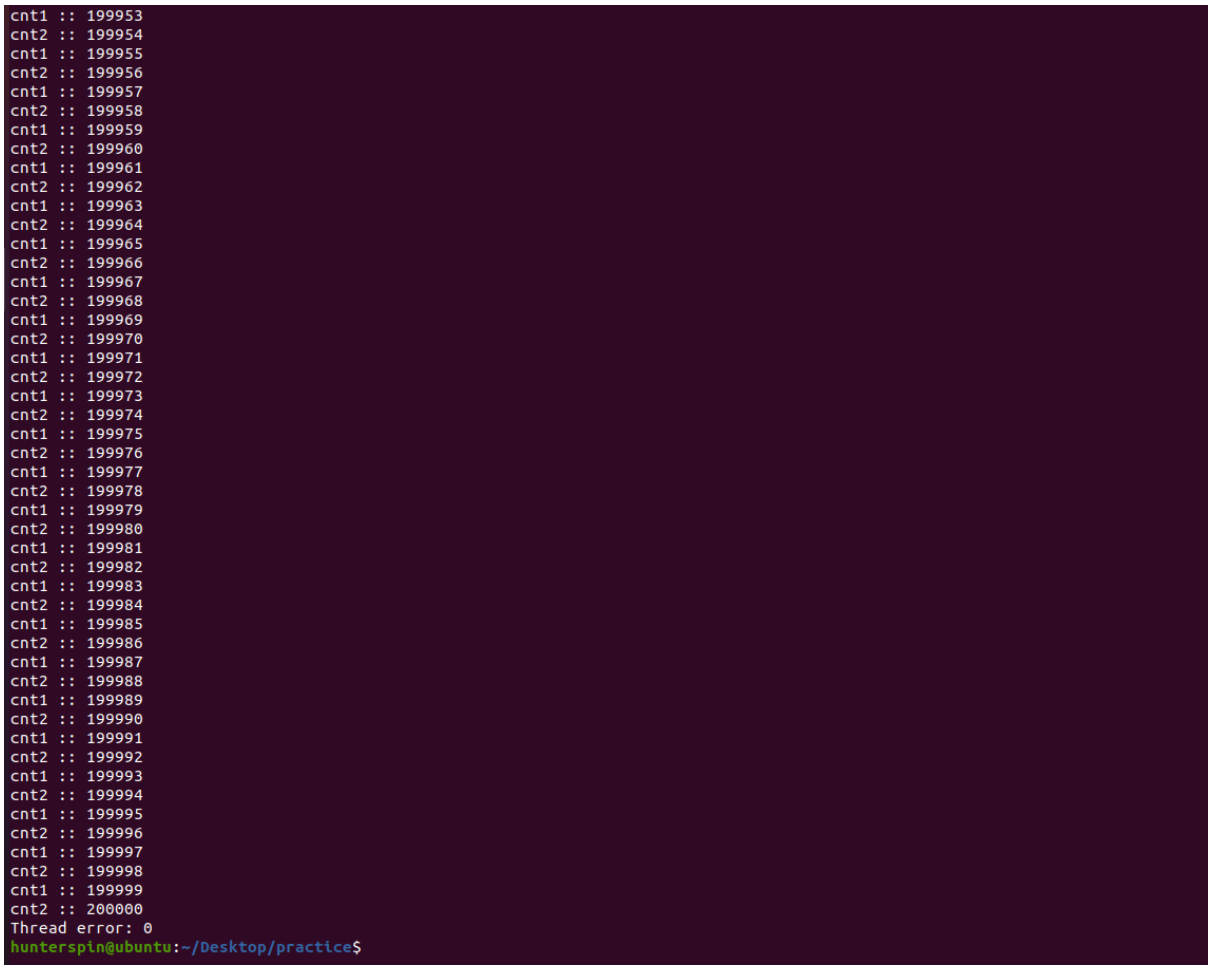
그후 값을 증가시키고 출력한 뒤 자신의 차례(flag[1])를 0으로 하여 자신의 차례가 끝남을 알린다. → 상대방이 while문에서 벗어나 자신의 차례를 진행한다.

두 함수 모두 공통적으로 자신의 할 일이 남았는데 상대방이 먼저 끝났을 경우flag가 항상 while문의 반대이기 때문에 할 일이 남은 함수는 계속해서 처리가 가능하다.

```
int main(){

	pthread_t thread1, thread2;
	int iret1, iret2;
	iret1 = pthread_create(&thread1, NULL, func0, NULL)
	iret2 = pthread_create(&thread2, NULL, func1, NULL);

	pthread_join(thread1, NULL);
	pthread_join(thread2, NULL);

	printf("Thread error: %d₩n", 200000 - cnt);

	exit(0);
}
```

마지막 메인문은 쓰레드를 두개 만들고 각각의 쓰레드마다 함수를 전달한 뒤 join을 통해 쓰레드가 종료되기를 기다린 후 제대로 실행됐을 경우 나와야 될 값(20만)에서 실제 나온 값을 뺀 값을 오류가 발생한 값이라고 출력한다.

실행 결과


cnt1 :: 199953
cnt2 :: 199954
cnt1 :: 199955
cnt2 :: 199956
cnt1 :: 199957
cnt2 :: 199958
cnt1 :: 199959
cnt2 :: 199960
cnt1 :: 199961
cnt2 :: 199962
cnt1 :: 199963
cnt2 :: 199964
cnt1 :: 199965
cnt2 :: 199966
cnt1 :: 199967
cnt2 :: 199968
cnt1 :: 199969
cnt2 :: 199970
cnt1 :: 199971
cnt2 :: 199972
cnt1 :: 199973
cnt2 :: 199974
cnt1 :: 199975
cnt2 :: 199976
cnt1 :: 199977
cnt2 :: 199978
cnt1 :: 199979
cnt2 :: 199980
cnt1 :: 199981
cnt2 :: 199982
cnt1 :: 199983
cnt2 :: 199984
cnt1 :: 199985
cnt2 :: 199986
cnt1 :: 199987
cnt2 :: 199988
cnt1 :: 199989
cnt2 :: 199990
cnt1 :: 199991
cnt2 :: 199992
cnt1 :: 199993
cnt2 :: 199994
cnt1 :: 199995
cnt2 :: 199996
cnt1 :: 199997
cnt2 :: 199998
cnt1 :: 199999
cnt2 :: 200000
Thread error: 0
hunterspin@ubuntu:~/Desktop/practice$