

운영체제

Thread 과제



SINCE 1947

서경대학교

SEOKYEONG UNIVERSITY



Thread 과제

-2019301005 김경민-

코드설명

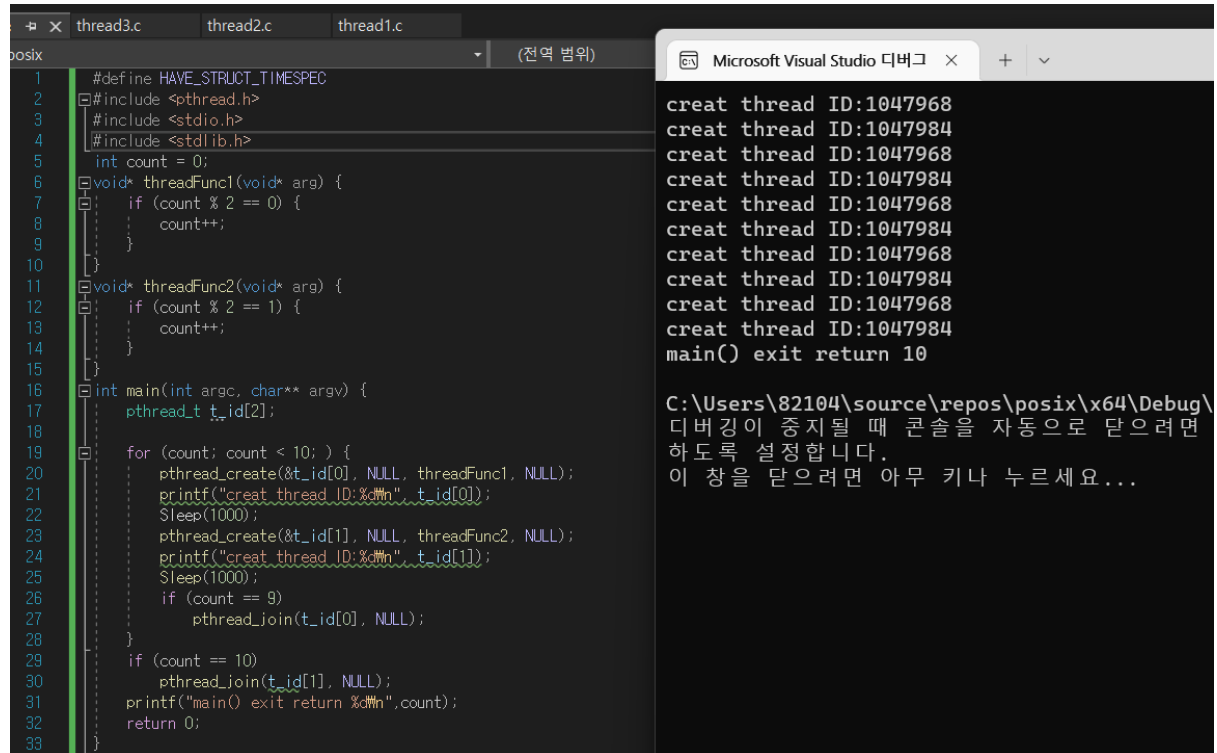
```
#define HAVE_STRUCT_TIMESPEC
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
int count = 0;
void* threadFunc1(void* arg) {
    if (count % 2 == 0) {
        count++;
    }
}
void* threadFunc2(void* arg) {
    if (count % 2 == 1) {
        count++;
    }
}
int main(int argc, char** argv) {
    pthread_t t_id[2];

    for (count; count < 10; ) {
        pthread_create(&t_id[0], NULL, threadFunc1, NULL);
        printf("creat thread ID:%d\n", t_id[0]);
        Sleep(1000);
        pthread_create(&t_id[1], NULL, threadFunc2, NULL);
        printf("creat thread ID:%d\n", t_id[1]);
        Sleep(1000);
        if (count == 9)
            pthread_join(t_id[0], NULL);
    }
    if (count == 10)
        pthread_join(t_id[1], NULL);
    printf("main() exit return %d\n", count);
    return 0;
}
```

여러 필요한 헤더 파일들을 정의 후 전역변수 count 생성 및 초기화
첫번째 Thread 함수는 count가 짝수라면 1증가, 2번째는 홀수라면 1증가
main에서 pthread t_id배열 생성 후 count가 10보다 작을때까지
pthread_create로 t_id[0]에는 첫번째 Thread 함수를 받아 생성하게 하고
t_id[1]에는 두번째 함수를 생성하게 하는데 1초마다 접근이 가능하게 함
if문을 사용하여 count가 9라면 t_id[0]를 종료, 10이라면 t_id[1]의 종료를 기다림
main함수가 종료됐음을 알리고 현재 count를 출력(10)
결과 화면은 1번에 수록

결과 화면

1.

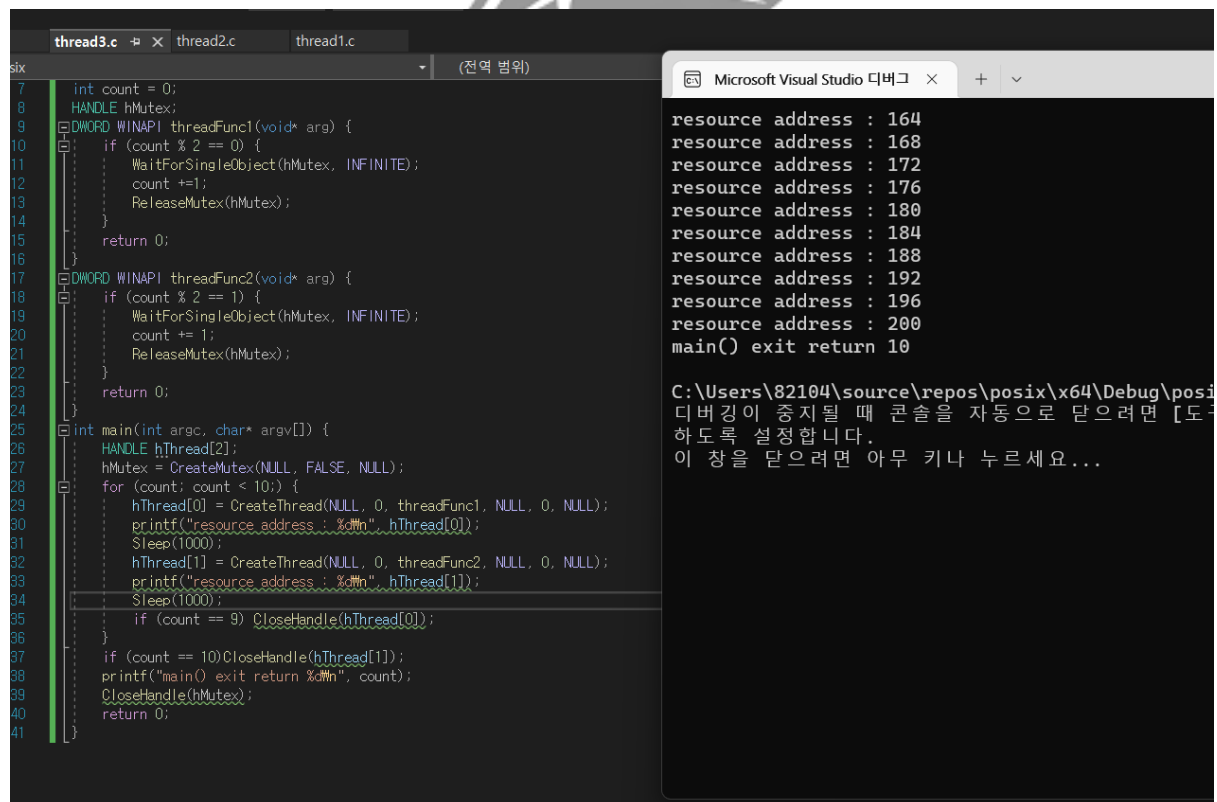


```
1  #define HAVE_STRUCT_TIMESPEC
2  #include <pthread.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  int count = 0;
6  void* threadFunc1(void* arg) {
7      if (count % 2 == 0) {
8          count++;
9      }
10 }
11 void* threadFunc2(void* arg) {
12     if (count % 2 == 1) {
13         count++;
14     }
15 }
16 int main(int argc, char** argv) {
17     pthread_t t_id[2];
18
19     for (count; count < 10; ) {
20         pthread_create(&t_id[0], NULL, threadFunc1, NULL);
21         printf("creat thread ID:%d\n", t_id[0]);
22         Sleep(1000);
23         pthread_create(&t_id[1], NULL, threadFunc2, NULL);
24         printf("creat thread ID:%d\n", t_id[1]);
25         Sleep(1000);
26         if (count == 9)
27             pthread_join(t_id[0], NULL);
28     }
29     if (count == 10)
30         pthread_join(t_id[1], NULL);
31     printf("main() exit return %d\n", count);
32     return 0;
33 }
```

```
creat thread ID:1047968
creat thread ID:1047984
creat thread ID:1047968
creat thread ID:1047984
creat thread ID:1047968
creat thread ID:1047984
creat thread ID:1047968
creat thread ID:1047984
creat thread ID:1047968
creat thread ID:1047984
main() exit return 10

C:\Users\82104\source\repos\posix\x64\Debug\posix
디버깅이 중지될 때 콘솔을 자동으로 닫으려면
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

2.



```
7  int count = 0;
8  HANDLE hMutex;
9  DWORD WINAPI threadFunc1(void* arg) {
10     if (count % 2 == 0) {
11         WaitForSingleObject(hMutex, INFINITE);
12         count += 1;
13         ReleaseMutex(hMutex);
14     }
15     return 0;
16 }
17 DWORD WINAPI threadFunc2(void* arg) {
18     if (count % 2 == 1) {
19         WaitForSingleObject(hMutex, INFINITE);
20         count += 1;
21         ReleaseMutex(hMutex);
22     }
23     return 0;
24 }
25 int main(int argc, char* argv[]) {
26     HANDLE hThread[2];
27     hMutex = CreateMutex(NULL, FALSE, NULL);
28     for (count; count < 10; ) {
29         hThread[0] = CreateThread(NULL, 0, threadFunc1, NULL, 0, NULL);
30         printf("resource address : %d\n", hThread[0]);
31         Sleep(1000);
32         hThread[1] = CreateThread(NULL, 0, threadFunc2, NULL, 0, NULL);
33         printf("resource address : %d\n", hThread[1]);
34         Sleep(1000);
35         if (count == 9) CloseHandle(hThread[0]);
36     }
37     if (count == 10) CloseHandle(hThread[1]);
38     printf("main() exit return %d\n", count);
39     CloseHandle(hMutex);
40     return 0;
41 }
```

```
resource address : 164
resource address : 168
resource address : 172
resource address : 176
resource address : 180
resource address : 184
resource address : 188
resource address : 192
resource address : 196
resource address : 200
main() exit return 10

C:\Users\82104\source\repos\posix\x64\Debug\posix
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

추가 시도

사실 처음에 좀 헤맸습니다. 그래서 여러가지 찾아보게 되었어서 적어보게 됐습니다. Mutex()라는 기능인데요, 예를들면 5-3의 카운트가 전역 변수인데요 이것을 1번 Thread함수가 접근하고 counting 하기전에 2번 Thread가 접근해버려서 Count+1 을 하는 Thread 함수가 멀티 쓰레딩 됐음에도 불구하고 결과가 2가 아닌 1이 되버리는 현상을 막기 위해서 Mutex를 사용함으로써 하나의 key를 주어서 1Thread가 접근했으면 2Thread가 대기하는 형식으로 만들어 1번이 접근 후 연산 후에 저장하고 2번이 접근하는 식으로도 만들어봤지만 인터넷에서도 거의 Linux나 Unix가 대부분이라서 사용법도 예제도 부족해서(Windows) 조금은 어렵고 완벽하게 이해는 못했지만 짜보기는 했습니다.

Windows에서 Mutex는 HANDLE 객체로 관리하게 됩니다. (전역 변수로 생성) HANDLE은 운영체제 내부의 리소스의 주소를 정수로 치환한 값입니다. (또 파일 개체, 프로세스 개체 등 많은 Win32 API 개체를 제어할 때 사용 형식)

또한 DWORD WINAPI도 Win32 API에서 제공하는 것입니다.(Type이 data형식) WaitForSingleObject로 hMutex에게 키를 주고 (한 객체 여러객체면 Multiple) INFINITE는 무한정으로 대기한다는 의미 입니다. TIMEOUT을 설정할 수 있습니다.

ReleaseMutex로 주었던 key를 다시 돌려 받습니다. 그렇게 Thread 함수를 만들고 HANDLE hThread 배열을 만들고 CreateMutex함수로 생성을 하게 되는데 CreateMutex(첫 인자로 Mutex의 보안속성 지정 일반적으로는 NULL을 하게 되고, 두번째 인자로는 생성과 동시에 실행권한 부여 여부인데 저는 false로 설정했습니다

마지막 인자는 Mutex의 이름을 지정하는 문자열) 이렇게 사용된다고 합니다. 그래서 각 hThread에 CreateThread로 1번 2번 Thread 함수를 생성해주고 9라면 hThread 1을 CloseHandle로 종료 10이라면 h thread2를 종료 시켜줍니다. 마지막에는 count를 출력하고 다시 CloseHandle로 생성한 hMutex 객체를 삭제 시켜줍니다. 이렇게 하면 1번 2번 Thread가 서로 각기 접근이 가능해서 충돌하는 일이 안생기게 만들고 Sleep을 사용하여 1초마다 접근하게 만들었습니다.

코드는 밑에 기입 했습니다. 결과 화면은 2번을 참조해주시면 감사하겠습니다.

```

#define HAVE_STRUCT_TIMESPEC
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <process.h>
int count = 0;
HANDLE hMutex;
DWORD WINAPI threadFunc1(void* arg) {
    if (count % 2 == 0) {
        WaitForSingleObject(hMutex, INFINITE);
        count +=1;
        ReleaseMutex(hMutex);
    }
    return 0;
}
DWORD WINAPI threadFunc2(void* arg) {
    if (count % 2 == 1) {
        WaitForSingleObject(hMutex, INFINITE);
        count += 1;
        ReleaseMutex(hMutex);
    }
    return 0;
}
int main(int argc, char* argv[]) {
    HANDLE hThread[2];
    hMutex = CreateMutex(NULL, FALSE, NULL);
    for (count; count < 10;) {
        hThread[0] = CreateThread(NULL, 0, threadFunc1,
NULL, 0, NULL);
        printf("resource address : %d\n", hThread[0]);
        Sleep(1000);
        hThread[1] = CreateThread(NULL, 0, threadFunc2,
NULL, 0, NULL);
        printf("resource address : %d\n", hThread[1]);
        Sleep(1000);
        if (count == 9) CloseHandle(hThread[0]);
    }
    if (count == 10)CloseHandle(hThread[1]);
    printf("main() exit return %d\n", count);
    CloseHandle(hMutex);
    return 0;
}

```

감사합니다.