

REPORT



과 목 :	네트워크프로그래밍
제출일자 :	2021. 10. 17
담당교수 :	황성호
학 과 :	컴퓨터공학과
학 번 :	201720970
이 름 :	권대한

1. ExCriticalSection 소스코드

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS

#include <windows.h>
#include <stdio.h>

#define MAXCNT 100000000 // 100,000,000
int g_count = 0;
CRITICAL_SECTION cs;

DWORD WINAPI MyThread1(LPVOID arg)
{
    for (int i = 0; i < MAXCNT; i++)
    {
        EnterCriticalSection(&cs);
        g_count += 2;
        LeaveCriticalSection(&cs);
    }
    return 0;
}

DWORD WINAPI MyThread2(LPVOID arg)
{
    for (int i = 0; i < MAXCNT; i++)
    {
        EnterCriticalSection(&cs);
        g_count -= 2;
        LeaveCriticalSection(&cs);
    }
    return 0;
}

int main(int argc, char* argv[])
{
    // 임계 영역 초기화
    InitializeCriticalSection(&cs);

    // 스레드 두 개 생성
    HANDLE hThread[2];
    hThread[0] = CreateThread(NULL, 0, MyThread1, NULL, 0, NULL);
    hThread[1] = CreateThread(NULL, 0, MyThread2, NULL, 0, NULL);

    // 스레드 두 개 종료 대기
    WaitForMultipleObjects(2, hThread, TRUE, INFINITE);
}
```

```

// 임계 영역 삭제
DeleteCriticalSection(&cs);

// 결과 출력
printf("g_count = %d\n", g_count);
return 0;
}

```

ExCriticalSection 실행결과

```

Microsoft Visual Studio 디버그 콘솔
g_count = 0
D:\OneDrive - 강원대학교\과제\3학년_2학기\네트워크 프로그래밍\C++ Projects\ExCriticalSection\ExCriticalSection\Debug\ExCriticalSection.exe(프로세스 18324개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...

```

2. ExEvent 소스코드

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS

#include <Windows.h>
#include <stdio.h>

#define BUFSIZE 10

HANDLE hReadEvent;
HANDLE hWriteEvent;
int buf[BUFSIZE];

DWORD WINAPI WriteThread(LPVOID arg)
{
    DWORD retval;

    for (int k = 1; k <= 500; k++)
    {
        // 읽기 완료 대기
        retval = WaitForSingleObject(hReadEvent, INFINITE);
        if (retval != WAIT_OBJECT_0)
        {
            break;
        }

        // 공유 버퍼에 데이터 저장
        for (int i = 0; i < BUFSIZE; i++)
        {
            buf[i] = k;
        }

        // 쓰기 완료 알림
        SetEvent(hWriteEvent);
    }

    return 0;
}

DWORD WINAPI ReadThread(LPVOID arg)
{
    DWORD retval;
```

```

while (1)
{
    // 쓰기 완료 대기
    retval = WaitForSingleObject(hWriteEvent, INFINITE);
    if (retval != WAIT_OBJECT_0)
    {
        break;
    }

    // 읽은 데이터 출력
    printf("Thread %4d: ", GetCurrentThreadId());
    for (int i = 0; i < BUFSIZE; i++)
    {
        printf("%3d ", buf[i]);
    }
    printf("\n");

    //버퍼 초기화
    ZeroMemory(buf, sizeof(buf));

    // 읽기 완료 알림
    SetEvent(hReadEvent);
}

return 0;
}

int main(int argc, char* argv[])
{
    // 자동 리셋 이벤트 두 개 생성(각각 비신호, 신호 상태)
    hWriteEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
    if (hWriteEvent == NULL)
    {
        return 1;
    }

    hReadEvent = CreateEvent(NULL, FALSE, TRUE, NULL);
    if (hReadEvent == NULL)
    {
        return 1;
    }
}

```

```

}

// 스레드 세 개 생성 W-R-R
HANDLE hThread[2];
hThread[0] = CreateThread(NULL, 0, WriteThread, NULL, 0, NULL);
hThread[1] = CreateThread(NULL, 0, ReadThread, NULL, 0, NULL);
hThread[2] = CreateThread(NULL, 0, ReadThread, NULL, 0, NULL);

// 스레드 세 개 종료 대기
WaitForMultipleObjects(3, hThread, TRUE, INFINITE);

// 이벤트 제거
CloseHandle(hWriteEvent);
CloseHandle(hReadEvent);

return 0;
}

```

ExEvent 실행결과

```

Thread 27308: 472 472 472 472 472 472 472 472 472 472
Thread 11064: 473 473 473 473 473 473 473 473 473 473
Thread 27308: 474 474 474 474 474 474 474 474 474 474
Thread 11064: 475 475 475 475 475 475 475 475 475 475
Thread 27308: 476 476 476 476 476 476 476 476 476 476
Thread 11064: 477 477 477 477 477 477 477 477 477 477
Thread 27308: 478 478 478 478 478 478 478 478 478 478
Thread 11064: 479 479 479 479 479 479 479 479 479 479
Thread 27308: 480 480 480 480 480 480 480 480 480 480
Thread 11064: 481 481 481 481 481 481 481 481 481 481
Thread 27308: 482 482 482 482 482 482 482 482 482 482
Thread 11064: 483 483 483 483 483 483 483 483 483 483
Thread 27308: 484 484 484 484 484 484 484 484 484 484
Thread 11064: 485 485 485 485 485 485 485 485 485 485
Thread 27308: 486 486 486 486 486 486 486 486 486 486
Thread 11064: 487 487 487 487 487 487 487 487 487 487
Thread 27308: 488 488 488 488 488 488 488 488 488 488
Thread 11064: 489 489 489 489 489 489 489 489 489 489
Thread 27308: 490 490 490 490 490 490 490 490 490 490
Thread 11064: 491 491 491 491 491 491 491 491 491 491
Thread 27308: 492 492 492 492 492 492 492 492 492 492
Thread 11064: 493 493 493 493 493 493 493 493 493 493
Thread 27308: 494 494 494 494 494 494 494 494 494 494
Thread 11064: 495 495 495 495 495 495 495 495 495 495
Thread 27308: 496 496 496 496 496 496 496 496 496 496
Thread 11064: 497 497 497 497 497 497 497 497 497 497
Thread 27308: 498 498 498 498 498 498 498 498 498 498
Thread 11064: 499 499 499 499 499 499 499 499 499 499
Thread 27308: 500 500 500 500 500 500 500 500 500 500

```