

Windows 操作系统

C/C++ 程序实验

姓名：_____陈展博_____

学号：_____1221001003_____

班级：_____计科 1 班_____

院系：_____信工_____

_____2024_____年__10__月__23__日

实验三 Windows 线程同步

一、背景知识

二、实验目的

在本实验中，通过对事件和互斥体对象的了解，来加深对 Windows 线程同步的理解。

- 1) 回顾系统进程、线程的有关概念，加深对 Windows 线程的理解。
- 2) 了解事件和互斥体对象。
- 3) 通过分析实验程序，了解管理事件对象的 API。
- 4) 了解在进程中如何使用事件对象。
- 5) 了解在进程中如何使用互斥体对象。
- 6) 了解父进程创建子进程的程序设计方法。

三、工具/准备工作

在开始本实验之前，请回顾教科书的相关内容。

您需要做以下准备：

- 1) 一台运行 Windows 操作系统的计算机。
- 2) 计算机中需安装 Visual Studio Code。

四、实验内容与步骤

1. 事件对象

步骤 1：登录进入 Windows 。

步骤 2：在“开始”菜单中单击“程序” - “Microsoft Visual Studio Code”，进入 Visual Studio 窗口。

步骤 3：新建项目名为“3-1”，并且新建项“3-1.cpp”。

步骤 4：在代码宏定义里添加 `#define _CRT_SECURE_NO_WARNINGS`

步骤 5：按“F5”开始调试，注意路径里不要含有中文。

步骤 6：按暂停按钮可暂停程序的执行，按终止按钮可终止程序的执行。



操作能否正常进行？如果不行，则可能的原因是什么？

操作能够正常运行。如果不行应该是文件格式问题或者是键入内容含有中文字符。

运行结果 (分行书写。如果运行不成功，则可能的原因是什么?)：

- 1) event created
- 2) child created
- 3) Parent waiting on child.
- 4) child process begining.....
- 5) event signaled
- 6) parent received the event signaling from child
- 7) Parent released.

这个结果与你期望的一致吗？(从进程并发的角度对结果进行分析)

这个结果与我期望的一致。父进程在 `void WaitForChild()` 函数中创建子进程并等待其发出信号，父进程阻塞在 `:: WaitForSingleObject(hEventContinue, INFINITE)` 等待，直到子进程发出信号，而当子进程在进入 `main` 函数时调用 `:: SignalParent()`；子进程打开了父进程创建的事件句柄，并设置了事件状态为信号状态，然后向父进程创建的事件对象发出信号。父进程的等待状态被子进程的信号解除，继续执行并输出相关信息。

阅读和分析程序 3-1，请回答：

- 1) 程序中，创建一个事件使用了哪一个系统函数？创建时设置的初始信号状态是什么？
 - a. 创建一个事件使用了 `:: CreateEvent` 函数。
 - b. 创建时设置的初始信号状态是 `FALSE` 非接受信号状态，非接受信号状态意味着等待该事件的进程将会被挂起，直到事件的状态被设置为信号状态。
- 2) 创建一个进程 (子进程) 使用了哪一个系统函数？

创建一个进程 (子进程) 使用了 `:: CreateProcess` 函数

- 3) 从步骤 6 的输出结果，对照分析 3-1 程序，可以看出程序运行的流程吗？请简单描述：

父进程先进入 `:: WaitForChild()` 函数创建事件对象，并调用 `:: CreateChild()` 来创建子进程，阻塞在 `:: WaitForSingleObject(hEventContinue, INFINITE)`；等待子进程通过事件对象发出信号。

父进程阻塞期间，子进程启动，进入 `:: SignalParent()` 函数向父进程创建的事件对象发出信号，在成功发出信号后，子进程结束，父进程在收到信号后继续运行直至释放。

选作：

- 1、程序 3-1 中删去所标注语句(选作:删去这句试试)，有何现象，试分析原因。

```
F:\Junior1\czb_os\3-1\3-1.exe X + v - □ X
event created
chlid created
Parent waiting on child.
child process begining.....
event signaledparent received the envent signaling from child
```

结果发现event signaled和parent received the envent signaling from child在同一行出现，因为父进程和子进程并发运行，移除sleep(1500)延时后，父进程几乎立即响应子进程发出的信号。在父进程收到信号时立即执行语句，导致在控制台缓冲区同时输出两条语句。

另外，现代操作系统和多核处理器可以同时运行多个线程，这可能导致两个进程的输出几乎同时发生出现这种情况。

2、如何修改程序，用两个控制台显示，并将结果与1对比说明。

```
// 返回的子进程的进程信息结构体
PROCESS_INFORMATION pi;
// 使用同一可执行文件和告诉它是一个子进程的命令行创建进程
BOOL bCreateOK = :: CreateProcess(
    szFilename,          // 生成的可执行文件名
    szCmdLine,           // 指示其行为与子进程一样的标志
    NULL,                // 子进程句柄的安全性
    NULL,                // 子线程句柄的安全性
    FALSE,               // 不继承句柄
    CREATE_NEW_CONSOLE,  // 特殊的创建标志 0
    NULL,                // 新环境
    NULL,                // 当前目录
    &si,                 // 启动信息结构
    &pi );               // 返回的进程信息结构
```

```
F:\Junior1\czb_os\3-1\3-1.exe X + v - □ X
event created
chlid created
Parent waiting on child.
parent received the envent signaling from child
Parent released.
请按任意键继续. . . |

F:\Junior1\czb_os\3-1\3-1.exe X + v - □ X
child process begining.....
event signaled
请按任意键继续. . . |
```

将创建子进程的系统函数里的特殊创建标志改为 CREATE_NEW_CONSOLE 即可用两个控制台显示。与 1 中出现顺序不变，不过结果在两个控制台上显示。

2. 互斥体对象

步骤 7：新建项目名为“3-2”，并且新建项“3-2.cpp”。

步骤 8：按“F5”开始调试，注意路径里不要含有中文。

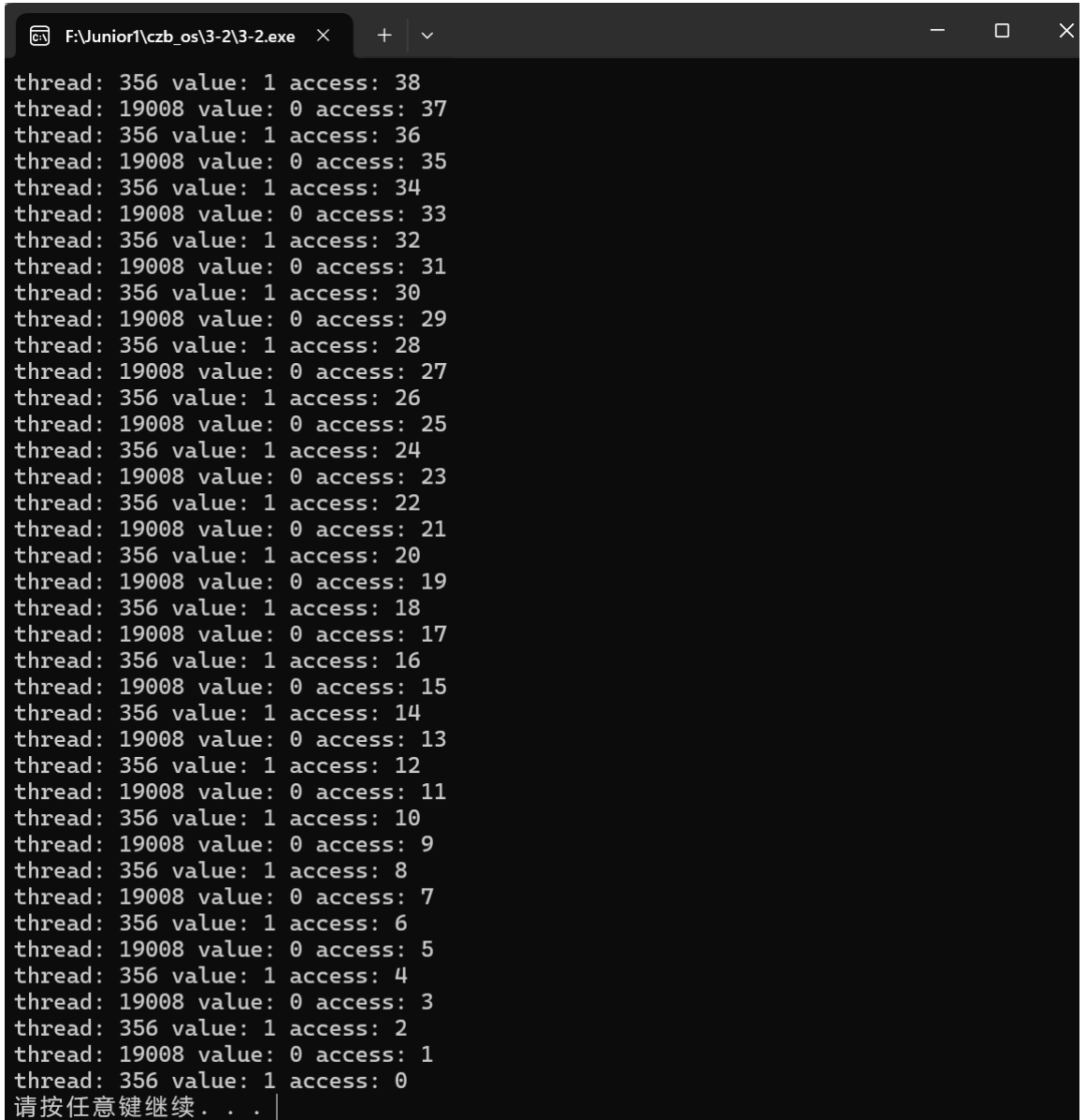
步骤 9：按暂停按钮可暂停程序的执行，按终止按钮可终止程序的执行。



操作能否正常进行？如果不行，则可能的原因是什么？

操作能够正常运行。如果不行应该是文件格式问题或者是键入内容含有中文字符。

- 1) 请描述运行结果 (如果运行不成功，则可能的原因是什么?) :



```
F:\Junior1\czb_os\3-2\3-2.exe
thread: 356 value: 1 access: 38
thread: 19008 value: 0 access: 37
thread: 356 value: 1 access: 36
thread: 19008 value: 0 access: 35
thread: 356 value: 1 access: 34
thread: 19008 value: 0 access: 33
thread: 356 value: 1 access: 32
thread: 19008 value: 0 access: 31
thread: 356 value: 1 access: 30
thread: 19008 value: 0 access: 29
thread: 356 value: 1 access: 28
thread: 19008 value: 0 access: 27
thread: 356 value: 1 access: 26
thread: 19008 value: 0 access: 25
thread: 356 value: 1 access: 24
thread: 19008 value: 0 access: 23
thread: 356 value: 1 access: 22
thread: 19008 value: 0 access: 21
thread: 356 value: 1 access: 20
thread: 19008 value: 0 access: 19
thread: 356 value: 1 access: 18
thread: 19008 value: 0 access: 17
thread: 356 value: 1 access: 16
thread: 19008 value: 0 access: 15
thread: 356 value: 1 access: 14
thread: 19008 value: 0 access: 13
thread: 356 value: 1 access: 12
thread: 19008 value: 0 access: 11
thread: 356 value: 1 access: 10
thread: 19008 value: 0 access: 9
thread: 356 value: 1 access: 8
thread: 19008 value: 0 access: 7
thread: 356 value: 1 access: 6
thread: 19008 value: 0 access: 5
thread: 356 value: 1 access: 4
thread: 19008 value: 0 access: 3
thread: 356 value: 1 access: 2
thread: 19008 value: 0 access: 1
thread: 356 value: 1 access: 0
请按任意键继续...
```

运行结果如图所示。其中 thread: 356 为 value+1 线程，thread: 19008 为 value-1 线程。

- 2) 根据运行输出结果，对照分析 3-2 程序，可以看出程序运行的流程吗？请描述：

可以，在 CCountUpDown 构造函数中，首先创建了一个互斥体，初始时由创建者拥有。CCountUpDown ud(50);限定了能够访问共享资源的次数，两个线程在启动时会尝试访问共享的 m_nValue。线程在执行时，每次访问前都必须等待获取互斥体的所有权，Inc 线程和 Dec 线程两个线程轮流访问，Inc 线程调用 IncThreadProc 函数，进入函数后通过 pThis ->

DoCount(+1);对共享资源进行修改操作，执行:: ReleaseMutex(m_hMutexValue);

之后将释放共享资源的控制权。

同理，Dec 线程调用 DecThreadProc 函数，进入函数后通过 `pThis->DoCount(-1);` 对共享资源进行修改操作，执行 `::ReleaseMutex(m_hMutexValue);`

之后将释放共享资源的控制权。

在 `m_nAccess` 可访问次数减为 0 后，两个线程完成工作，进程结束。