操作系统实验指导

实验课题：

典型同步问题模拟处理编程设计与实现

翟高寿

北京交通大学计算机学院

2021 年 2 月修订

1、实验目的

探索、理解并掌握操作系统同步机制的应用编程方法，针对典型的同步问题，构建基于Windows（或 Linux）操作系统同步机制的解决方案。

2、实验内容

了解、熟悉和运用 Windows（或 Linux）操作系统同步机制及编程方法，针对典型的同步问题，譬如生产者-消费者问题、读者优先的读者-写者问题、写者优先的读者-写者问题、读者数限定的读者-写者问题、哲学家就餐问题等（任选四个即可），编程模拟实现相应问题的解决方案。

3、实验要求

典型同步问题模拟处理编程功能设计要求为：运用 Windows（或 Linux）操作系统同步机制，从生产者-消费者问题、读者优先的读者-写者问题、写者优先的读者-写者问题、读者数限定的读者-写者问题、哲学家就餐问题等典型同步问题中选取四个同步问题，分析、设计和编程模拟实现相应问题的解决方案。

实验报告撰写和提交要求：

（1）实验报告内容，须涵盖开发环境、运行环境、测试环境、源程序文件及源码清单、实验步骤、同步问题的规范描述、技术难点及解决方案、关键数据结构和算法流程、编译运行测试过程及结果截图、疑难解惑及经验教训、结论与体会等；

（2）在实验报告内容（如运行结果截图等适当位置）中应有机融入个人姓名、学号、计算机系统信息等凸显个人标记特征的信息；

（3）实验报告文档提交格式可为 Word 文档、WPS 文档或 PDF 文档。

4、成绩评价说明

本实验课题成绩评价满分按 5 分计。

实验课题得分根据自我独立完成情况、完成质量及实验报告水平综合决定。一般来说，获得满分要求有明确一致多项证据证实自我独立完成且满足实验课题所有要求。相反地，若

无明确一致证据证实自我独立完成、甚至有明确证据证实存在抄袭行为，则酌情减分直至降为零分。

成绩评定细则指导建议如下：

（1）2 分：第一个同步问题的正确编程解决方案为 2 分。

（2）1 分*3：其余三个同步问题的正确编程解决方案分别各计 1 分。

（3）计算上述两项得分之和作为本实验课题成绩。

（4）互评成绩结果在提交慕课平台时按四舍五入取整处理。

5、实验编程提示-Windows 线程操作及同步机制相关函数

Windows 线程操作及同步机制所涉头文件包含语句、函数调用语句或函数原型如下：

（1）#include <windows.h>

（2）线程函数原型及框架

```
DWORD WINAPI ThreadExecutiveZGS(LPVOID lpParameter)
{
    int *pID = (int*)lpParameter;
    ……
    return 0;
}
```

（3）线程创建所涉相关变量及函数调用语句

```
HANDLE hThread[2];
int nPID0 = 0, nPID1 = 1;

if ((hThread[0] = CreateThread(NULL, 0, ThreadExecutiveZGS, &nPID0, 0, NULL)) == NULL)
{
    printf("线程  ThreadExecutiveZGS-0  创建失败！\n");
    exit(0);
}
if ((hThread[1] = CreateThread(NULL, 0, ThreadExecutiveZGS, &nPID1, 0, NULL)) == NULL)
{
    printf("线程  ThreadExecutiveZGS-1  创建失败！\n");
    exit(0);
}
```

（4）等待线程函数原型

```
DWORD WaitForMultipleObjects(
  DWORD nCount,
  CONST HANDLE *lpHandles,
  BOOL fWaitAll,
  DWORD dwMilliseconds
);
```

在主函数中调用本函数，以用来实现主线程对两个银行账户转账线程的等待：

```
WaitForMultipleObjects(2, hThread, TRUE, INFINITE);
```

（5）关于互斥信号量创建的函数原型及应用示例

```
HANDLE CreateMutex(
  LPSECURITY_ATTRIBUTES lpMutexAttributes,
  BOOL bInitialOwner,
  LPCTSTR lpName
);

HANDLE hMutex = CreateMutex(NULL, FALSE, "MutexToProtectCriticalResource");
```

（6）关于互斥信号量释放（开锁/唤醒）的函数原型及应用示例

```
BOOL ReleaseMutex(
  HANDLE hMutex
);

ReleaseMutex(hMutex);
```

（7）关于信号量创建的函数原型及应用示例

```
HANDLE WINAPI CreateSemaphore(
  _In_opt_ LPSECURITY_ATTRIBUTES  lpSemaphoreAttributes, //安全属性
  _In_     LONG lInitialCount, //初始数值
  _In_     LONG lMaximumCount, //信号量的最大值
  _In_opt_ LPCWSTR lpName //信号量名字
);

HANDLE g_hEmpty =NULL;
g_hEmpty = CreateSemaphore(NULL, 1, 10, "Empty");
```

（8）关于信号量打开的函数原型及应用示例

```
HANDLE WINAPI OpenSemaphore(
   _In_ DWORD dwDesiredAccess,
   _In_ BOOL bInheritHandle,
   _In_ LPCSTR lpName
);

g_hEmpty = OpenSemaphore(SEMAPHORE_ALL_ACCESS, TRUE, "Empty");
```

（9）关于信号量释放（唤醒）的函数原型及应用示例

```
BOOL WINAPI ReleaseSemaphore(
   _In_ HANDLE hSemaphore,
   _In_ LONG lReleaseCount,
   _Out_opt_ LPLONG lpPreviousCount
);

ReleaseSemaphore(g_hEmpty, 1, NULL);
```

（10）关于信号量申请（等待）的函数原型及应用示例

```
DWORD WaitForSingleObject(
   HANDLE hHandle,
   DWORD dwMilliseconds
);

WaitForSingleObject(hMutex, INFINITE);
WaitForSingleObject(g_hEmpty, INFINITE);
```

（12）关于线程挂起的函数原型及应用示例

```
VOID Sleep(DWORD dwMilliseconds);

Sleep(1000);
```

（13）关于线程或信号量句柄关闭的函数原型及应用示例

```
BOOL CloseHandle(HANDLE hObject);

CloseHandle(hThread[0]);
CloseHandle(g_hEmpty);
```

6、实验编程提示-Linux 线程操作及同步机制相关函数

Linux 线程操作及同步机制所涉头文件包含语句、函数调用语句或函数原型如下：

（1）#include <pthread.h>

（2）编译命令示例（末尾须加上选项-lpthread）

gcc -o ThreadCSExclusion ThreadCriticalSectionExclusion.c **-lpthread**

（3）线程函数原型及框架示例

```
void* ThreadExecutiveZGS(void* zThreadName)
{
    char *pThreadName = (char*)zThreadName;
    ……
    return (void *)0;
}
```

（4）线程创建所涉相关函数原型及应用示例

```
int pthread_creat(pthread_t *tid, const pthread_attr_t *attr, void*(*start_routine)(void *), void *arg);
```

返回值：成功返回 0，错误返回错误号

第一个参数 tid：线程标识符（输出型参数）

第二个参数 attr：线程属性，一般设置为 NULL（表示线程属性取缺省值）

第三个参数 start_routine：函数指针，指向新线程即将执行的代码

第四个参数 arg：新线程对应函数的参数序列封装结果

```
pthread_t tid1;
if (pthread_create(&tid1, NULL, ThreadExecutiveZGS, "thread1"))
{
    printf("线程  ThreadExecutiveZGS-1  创建失败！\n");
    exit(0);
}
```

（5）关于等待线程的函数原型及应用示例

```
int pthread_join(pthread_t tid, void **rval_ptr);
```

返回值：成功返回 0，错误返回错误编号

第一个参数 tid：被等待线程的标识符（输入型参数）

第二个参数 rval_ptr：指向被等待线程对应函数的返回值

在主函数中调用本函数，以用来实现主线程对指定银行账户转账线程的等待：

```
void *ret1;
pthread_join(tid1, &ret1);
```

（6）关于互斥锁初始化、上锁/开锁的函数原型及应用示例

```
int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr);
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

/*上锁*/
if (pthread_mutex_lock(&mutex) != 0)
{
    printf("上锁失败！\n");
    exit(1);
}

/*开锁*/
if (pthread_mutex_unlock(&mutex) != 0)
{
    printf("开锁失败！\n ");
    exit(1);
}
```

（7）关于信号量初始化、申请、释放等操作的函数原型及应用示例

```
#include <semaphore.h>

int sem_init(sem_t *sem, int pshared, unsigned int value);
```

// sem 为信号量，pshared 用于区分进程/线程间共享（0 为当前进程各线程间共享），value 为信号量初始值。

```
int sem_wait(sem_t *sem); //信号量申请操作
int sem_post(sem_t *sem); //信号量释放操作
int sem_destroy(sem_t *sem); //信号量销毁操作
```

```
sem_t sem;
sem_init(&sem,0,10);
sem_wait(&sem);
sem_post(&sem);
```

5、国产平台鼓励说明

鼓励基于华为 OpenEuler 操作系统、龙芯 Loongson 操作系统等国产操作系统开展本实验课题的设计实现和测试验证，实验课题成绩及平时成绩评定将给予适当升档处理。对于北京交通大学的同学，可申请操作系统课程组华为泰山服务器（OpenEuler 操作系统）账号，亦可自主申请华为云虚拟机搭建 OpenEulerOS 等国产操作系统平台完成本实验课题。