

操作系统课程设计实验报告

实验名称： 生产者消费者问题

姓名/学号： 卜梦煜/1120192419

一、 实验目的

- 理解生产者消费者问题，在不同操作系统中解决生产者消费者问题。
- 掌握不同系统中创建多个进程、进行进程间通信、实现进程间同步的原理、方法、以及相关的系统 API 函数。

二、 实验内容

- 一个大小为 3 的缓冲区，初始为空
- 2 个生产者
 - 随机等待一段时间，往缓冲区添加数据
 - 若缓冲区已满，等待消费者取走数据再添加
 - 重复 6 次
- 3 个消费者
 - 随机等待一段时间，从缓冲区读取数据
 - 若缓冲区为空，等待生产者添加数据后再读取
 - 重复 4 次

说明：

- 显示每次添加和读取数据的时间及缓冲区里的数据（指缓冲区里的具体内容）
- 生产者和消费者用进程模拟（不能用线程）
- Linux/Window 下都需要做

三、 实验环境

Windows: Dev-C++ 5.7.1.0、命令行

Linux: Ubuntu 18.04.3、VSCode 1.62

四、 程序设计与实现

1. Windows 部分

(1) 程序设计

主进程：

1) 利用 `CreateFileMapping()` 函数创建文件映射对象作为各进程通信的共享区，对第一个参数赋值 `INVALID_HANDLE_VALUE`，从而使该文件对象为临时文件。

2) 利用 `OpenFileMapping()` 打开临时文件映射对象，利用 `MapViewOfFile()` 将临时文件映射对象的一个视口映射到主进程。

3) 对公共区做初始化操作。包括令缓冲区的头尾指针指向 0，利用 `CreateSemaphore()` 创建信号量 `full`、`empty`、`mutex`。

4) 利用 `UnmapViewOfFile()` 函数从主进程中解除临时文件映射对象的映射，利用 `CloseHandle()` 关闭主进程打开的临时文件映射对象句柄。

5) 利用 `CreateProcess()` 函数克隆主进程 5 次，作为 2 个生产者进程、3 个消费者进程，将自定义的进程编号通过命令行参数传入子进程。

6) 利用 `WaitForSingleObject()` 函数等待 5 个子进程结束，利用 `CloseHandle()` 函数关闭 5 个子进程。

7) 利用 `CloseHandle()` 删除临时文件映射对象，结束主进程。

生产者进程：

1) 利用 `OpenFileMapping()` 打开临时文件映射对象，利用 `MapViewOfFile()` 将临时文件映射对象的一个视口映射到当前进程。

2) 对公共区做初始化操作。打开公共区的信号量 `full`、`empty`、`mutex`。

3) 利用 `Sleep()` 随机休眠一段时间。

4) 利用 `WaitForSingleObject()` 函数依次做 `P(empty)`、`P(mutex)` 操作。

5) 获取当前时间，向共享区队尾写入一个随机字母。

6) 打印当前时间及缓冲区中内容。

7) 利用 `ReleaseSemaphore()` 函数释放信号量，即 `V(full)`、`V(mutex)`。

8) 重复 3) 至 7) 步 6 次。

9) 利用 `CloseHandle()` 关闭信号量、利用 `UnmapViewOfFile()` 函数从当前进程中解除临时文件映射对象的映射, 利用 `CloseHandle()` 关闭当前进程打开的临时文件映射对象句柄。

消费者进程:

1) 利用 `OpenFileMapping()` 打开临时文件映射对象, 利用 `MapViewOfFile()` 将临时文件映射对象的一个视口映射到当前进程。

2) 对公共区做初始化操作。打开公共区的信号量 `full`、`empty`、`mutex`。

3) 利用 `Sleep()` 随机休眠一段时间。

4) 利用 `WaitForSingleObject()` 函数依次做 `P(full)`、`P(mutex)` 操作。

5) 获取当前时间, 从共享区队首取出一个字母。

6) 打印当前时间及缓冲区中内容。

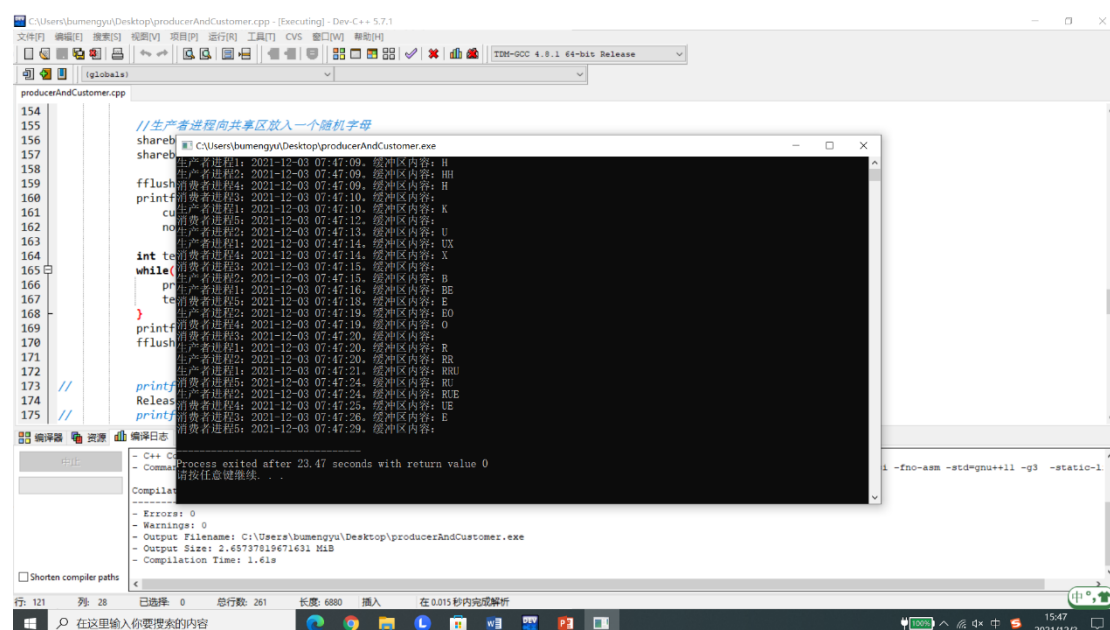
7) 利用 `ReleaseSemaphore()` 函数释放信号量, 即 `V(empty)`、`V(mutex)`。

8) 重复 3) 至 7) 步 4 次。

9) 利用 `CloseHandle()` 关闭信号量、利用 `UnmapViewOfFile()` 函数从当前进程中解除临时文件映射对象的映射, 利用 `CloseHandle()` 关闭当前进程打开的临时文件映射对象句柄。

(2) 运行结果

Windows 运行结果如下图所示:



```
154 //生产者进程向共享区放入一个随机字母
155 shareb
156
157 //消费者进程从共享区取出一个随机字母
158
159 fflush
160 printf
161 //消费者进程1: 2021-12-03 07:47:10, 缓冲区内容: K
162 //消费者进程5: 2021-12-03 07:47:12, 缓冲区内容: U
163 //生产者进程1: 2021-12-03 07:47:14, 缓冲区内容: UX
164 //消费者进程4: 2021-12-03 07:47:15, 缓冲区内容: X
165 //消费者进程3: 2021-12-03 07:47:15, 缓冲区内容: B
166 //生产者进程2: 2021-12-03 07:47:15, 缓冲区内容: BE
167 //消费者进程5: 2021-12-03 07:47:18, 缓冲区内容: E
168 //生产者进程4: 2021-12-03 07:47:19, 缓冲区内容: EO
169 //消费者进程3: 2021-12-03 07:47:20, 缓冲区内容: RR
170 //生产者进程1: 2021-12-03 07:47:20, 缓冲区内容: R
171 //生产者进程2: 2021-12-03 07:47:21, 缓冲区内容: RRU
172 //消费者进程5: 2021-12-03 07:47:24, 缓冲区内容: RU
173 //生产者进程2: 2021-12-03 07:47:24, 缓冲区内容: RUE
174 //消费者进程4: 2021-12-03 07:47:25, 缓冲区内容: UE
175 //消费者进程3: 2021-12-03 07:47:26, 缓冲区内容: E
176 //消费者进程5: 2021-12-03 07:47:29, 缓冲区内容: E
```

分析结果可知, 生产者、消费者通过信号量同步缓冲区中字符数以及互斥访

间缓冲区。满足缓冲区大小为 3、2 个生产者分别写入 6 个字母、3 个消费者分别取出 4 个字母的要求。

2. Linux 部分

(1) 程序设计

主进程：

- 1) 利用 `shmget()` 创建共享内存。
- 2) 利用 `semget()` 创建 3 个信号量存入 `semid` 中，信号量依次为 `full`、`empty`、`mutex`，并利用 `semctl()` 为信号量初始化。
- 3) 利用 `shmat()` 将共享内存链接到主进程，对缓冲区变量初始化，使头尾指针指向 0 位置。
- 4) 利用 `fork()` 创建 2 个生产者进程、3 个消费者进程。
- 5) 等待子进程运行结束。利用 `shmdt()` 函数解除主进程与共享内存的绑定。利用 `shmctl()` 删除共享内存，其中第二个参数为 `IPC_RMID`。利用 `semctl()` 函数循环删除 3 个信号量。

生产者进程：

- 1) 利用 `shmat()` 函数将共享内存链接到当前进程。
- 2) 利用 `Sleep()` 函数随机等待一段时间。
- 3) 利用 `semop()` 函数依次进行 `P(empty)`、`P(mutex)` 操作。
- 4) 获取当前时间、向共享区队尾写入一个随机字母。
- 5) 打印当前时间和缓冲区中数据。
- 6) 利用 `semop()` 函数执行 `V(mutex)`、`V(full)` 操作。
- 7) 重复 2 至 6 步 6 次。
- 8) 利用 `shmdt()` 解除进程与共享内存的绑定。

消费者进程：

- 1) 利用 `shmat()` 函数将共享内存链接到当前进程。
- 2) 利用 `Sleep()` 函数随机等待一段时间。
- 3) 利用 `semop()` 函数依次进行 `P(full)`、`P(mutex)` 操作。

- 4) 获取当前时间、从共享区队首取出一个字母。
- 5) 打印当前时间和缓冲区中数据。
- 6) 利用 `semop()` 函数执行 `V(mutex)`、`V(empty)` 操作。
- 7) 重复 2 至 6 步 6 次。
- 8) 利用 `shmdt()` 解除进程与共享区的绑定。

(2) 运行结果

Linux 下运行结果如下图所示：

```
bumengyu@bumengyu-virtual-machine: ~/VS Code/Code/OS-EX-3
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
bumengyu@bumengyu-virtual-machine:~/VS Code/Code/OS-EX-3 ./producerAndConsumer生产者进程2: Frl Dec 3 13:53:13 2021
缓冲区内容: V
生产者进程1: Frl Dec 3 13:53:13 2021
缓冲区内容: VE
消费者进程2: Frl Dec 3 13:53:14 2021
缓冲区内容: E
生产者进程1: Frl Dec 3 13:53:14 2021
缓冲区内容: V
消费者进程1: Frl Dec 3 13:53:15 2021
缓冲区内容: V
生产者进程2: Frl Dec 3 13:53:15 2021
缓冲区内容: E
生产者进程1: Frl Dec 3 13:53:17 2021
缓冲区内容: Y
消费者进程2: Frl Dec 3 13:53:17 2021
缓冲区内容: E
生产者进程2: Frl Dec 3 13:53:18 2021
缓冲区内容: V
消费者进程2: Frl Dec 3 13:53:18 2021
缓冲区内容: Y
生产者进程1: Frl Dec 3 13:53:19 2021
缓冲区内容: YV
生产者进程2: Frl Dec 3 13:53:19 2021
缓冲区内容: E
消费者进程3: Frl Dec 3 13:53:20 2021
缓冲区内容: VK
生产者进程2: Frl Dec 3 13:53:20 2021
缓冲区内容: VKD
消费者进程1: Frl Dec 3 13:53:21 2021
缓冲区内容: KD
生产者进程1: Frl Dec 3 13:53:21 2021
缓冲区内容: DD
消费者进程2: Frl Dec 3 13:53:21 2021
缓冲区内容: DD
生产者进程2: Frl Dec 3 13:53:24 2021
缓冲区内容: DD
消费者进程3: Frl Dec 3 13:53:25 2021
缓冲区内容: DD
生产者进程1: Frl Dec 3 13:53:25 2021
缓冲区内容: DD
消费者进程2: Frl Dec 3 13:53:26 2021
缓冲区内容: DD
消费者进程1: Frl Dec 3 13:53:27 2021
缓冲区内容: D
消费者进程2: Frl Dec 3 13:53:31 2021
缓冲区内容: D
```

分析结果可知，生产者、消费者通过信号量同步缓冲区中字符数，互斥访问缓冲区。满足缓冲区大小为 3、2 个生产者分别写入 6 个字母、3 个消费者分别

取出 4 个字母的要求。

五、实验收获与体会

通过实验三，我学习了 Windows 和 Linux 中与进程创建、进程通信、进程同步互斥有关的系统 API。

对 Windows 内存管理器，使用创建文件映射对象的方式进行进程通信，最常用的是用 `INVALID_HANDLE_VALUE` 参数创建的临时文件映射对象，其优点是不需要创建实际的文件，方便进程之间大量通信；由于文件映射对象为可被不同进程访问，因此只要将文件映射对象的一个视图映射到对应进程即可实现对共享区的访问。Windows 信号量通过 `CreateSemaphore()` 创建，`WaitForSingleObject()` 相当于 P 操作，`ReleaseSemaphore()` 相当于 V 操作，通过 `CloseHandle()` 关闭信号量。

对 Linux，使用共享内存的方式进行进程间通信。使用 `shmget()` 创建共享内存，进程只要用 `shmat()` 将其附加到自己的地址空间即可进行读写操作。子进程退出时，使用 `shmdt()` 断开与共享内存的链接，主进程退出时，使用 `shmctl()` 删除进程。Linux 信号量通过 `semget()` 创建，`semop()` 进行 P、V 操作，对信号量值的具体操作由结构体 `sembuf` 中 `sem_op` 决定，信号量通过 `semctl()` 关闭。