**БУ ВО Ханты-Мансийского автономного округа – Югры**

**«Сургутский государственный университет»**

**Политехнический институт**

**Кафедра автоматики и компьютерных систем**

**ОТЧЁТ**

**по лабораторной работе №5**

**по дисциплине: «Операционные системы»**

**Выполнил: студент группы №609-21**

**Шумилов И. Д.**

**Принял: старший преподаватель кафедры АиКС**

**Кривицкая М. А.**

**Сургут**

**2024г.**

**Цель работы:** изучить принципы синхронизации процессов и потоков, функции, структуры данных и объекты ядра, использующиеся для синхронизации в пользовательском режиме и в режиме ядра, и получить навыки практического использования прикладного программного интерфейса.

**Задание на лабораторную работу:** модифицировать и отладить данную программу так, чтобы задача решалась корректно при любом количестве производителей и потребителей и их произвольной относительной скорости работы. При решении задачи должен использоваться весь исходный код программы. Решение считается приемлемым, если в ходе работы программы потоки получают монопольный доступ к буферу и не появляются сообщения о считывании неверных значений и потери информации.

**Ход работы:**

*Листинг 1. Глобальные переменные, объявленные для реализации задачи:*

*HANDLE hMutex; /\* Global Mutex \*/*

*HANDLE hSemaphoreWr, hSemaphoreRd;*

*Листинг 2. Модифицированная версия главной функции:*

*int main() {*

*srand((unsigned)time(NULL));*

*Buffer\* Buf = Buffer::CreateBuffer(BufferSize); /\*Создание буфера\*/*

*HANDLE hThreads[cProducers + cConsumers];*

*/\* Вспомогательный поток, ожидающий нажатие клавиши \*/*

*CreateThread(0, 0, getkey, 0, 0, 0);*

*/\* Создание мьютекса \*/*

*hMutex = CreateMutexW(NULL, FALSE, NULL);*

*hSemaphoreRd = CreateSemaphoreW(NULL, 0, BufferSize, NULL);*

*hSemaphoreWr = CreateSemaphoreW(NULL, BufferSize, BufferSize, NULL);*

*if (hMutex == NULL || hSemaphoreRd == NULL || hSemaphoreWr == NULL)*

*return 1;*

*/\* Создание потоков-производителей \*/*

*for (int i = 0; i < cProducers; i++) {*

*hThreads[i] = CreateThread(0, 0, producer, Buf, 0, 0);*

*if (hThreads[i] == NULL)*

*return 1;*

*}*

*/\* Создание потоков-потребителей \*/*

*for (int i = cProducers; i < cProducers + cConsumers; i++) {*

*hThreads[i] = CreateThread(0, 0, consumer, Buf, 0, 0);*

*if (hThreads[i] == NULL)*

*return 1;*

*}*

*WaitForMultipleObjects(cProducers + cConsumers, hThreads, true, INFINITE);*

*/\* Закрытие всех открытых дескрипторов \*/*

*for (int i = 0; i < cProducers + cConsumers; i++)*

*CloseHandle(hThreads[i]);*

*CloseHandle(hMutex);*

*cout << "Threads finished" << endl;*

*cin.get();*

*return 0;*

*}*

*Листинг 3. Модифицированная версия функции-потока производителя:*

*/\* Поток-производитель \*/*

*DWORD \_\_stdcall producer(void\* b) {*

*DWORD dwWaitResult;*

*while (cOperations > 0) {*

*dwWaitResult = WaitForSingleObject(hSemaphoreWr, 0);*

*switch (dwWaitResult) {*

*case WAIT\_OBJECT\_0:*

*dwWaitResult = WaitForSingleObject(hMutex, INFINITE);*

*int item;*

*switch (dwWaitResult) {*

*case WAIT\_OBJECT\_0:*

*item = rand();*

*((Buffer\*)b)->PutItem(item);*

*cout << "Put item: " << item << endl;*

*cOperations--;*

*ReleaseSemaphore(hSemaphoreRd, 1, NULL);*

*Sleep(500 + rand() % 100);*

*ReleaseMutex(hMutex);*

*break;*

*case WAIT\_ABANDONED:*

*return 0;*

*}*

*break;*

*case WAIT\_ABANDONED:*

*return 0;*

*}*

*}*

*return 0;*

*}*

*Листинг 4. Модифицированная версия функции-потока потребителя:*

*/\* Поток-потребитель \*/*

*DWORD \_\_stdcall consumer(void\* b) {*

*DWORD dwWaitResult;*

*while (cOperations > 0) {*

*dwWaitResult = WaitForSingleObject(hSemaphoreRd, 0);*

*switch (dwWaitResult) {*

*case WAIT\_OBJECT\_0:*

*dwWaitResult = WaitForSingleObject(hMutex, INFINITE);*

*switch (dwWaitResult) {*

*case WAIT\_OBJECT\_0:*

*cout << "Get item: " << ((Buffer\*)b)->GetItem() << endl;*

*// ((Buffer\*)b)->GetItem();*

*cOperations--;*

*ReleaseSemaphore(hSemaphoreWr, 1, NULL);*

*Sleep(500 + rand() % 100);*

*ReleaseMutex(hMutex);*

*break;*

*case WAIT\_ABANDONED:*

*return 0;*

*}*

*break;*

*case WAIT\_ABANDONED:*

*return 9;*

*}*

*}*

*return 0;*

*}*

**Вывод:** были изучены принципы синхронизации процессов и потоков, функции, структуры данных и объекты ядра, использующиеся для синхронизации в пользовательском режиме и в режиме ядра, и получить навыки практического использования прикладного программного интерфейса.