МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

**Домашнее задание**

|  |  |  |
| --- | --- | --- |
| по дисциплине | Операционные системы |  |

|  |  |
| --- | --- |
| Студента | Абрамова Егора Владимировича |
|  | фамилия, имя, отчество полностью |
| Курс | 1 Группа ФИТ-221 |
| Направление | 02.03.02 Фундаментальная информатика |
|  | и информационные технологии |
|  | код, наименование |
| Руководитель | Старший преподаватель |
|  | должность, ученая степень, звание |
|  | Карабцов Р. Д. |
|  | фамилия, инициалы |
| Выполнил | 19.04.2024 |
|  | дата, подпись студента |
| баллы |  |
|  | дата, подпись руководителя |

Омск-2024

**Задание.** Разработать программы в ОС Windows для двух отдельных процессов, использующих общую память. В первой программе должна создаваться разделяемая память и семафор для взаимодействия между процессами. Во второй открывается доступ к этой же разделяемой памяти и семафору. Первая программа должна после задержки в 10 - 15 секунд записать какой-то текст в разделяемую память и указать его готовность с помощью семафора, а вторая программа должна вначале прочитать текстовую информацию из того места разделяемой памяти, где она должна появиться и выдать полученную информацию на экран с примечанием о существе действия, затем перейти к ожиданию разрешающего значения семафора и только после его завершения выдать на экран содержимое из разделяемой памяти с соответствующим примечанием. Вместо семафора в Windows можно использовать событие или мьютекс. Обе программы должны после получения действующего указателя на область общей памяти вывести адрес этой области в консольное окно с соответствующими пояснениями. Первая программа, кроме того, после формирования данных для второго процесса, должна запросить 1000 байтов дополнительной памяти с помощью универсальной функции VirtualAlloc и записать в эту область памяти, начиная с ее начала, сколько возможно символов латинского алфавита, записывая эти символы, пропуская по 399 свободных байтов (т.е. записывая 'a' в байт с нулевым смещением этой области памяти, затем 'b' в байт со смещением 400, далее 'c' в байт со смещением 800 и т.д., выполняя такое продвижение на 400 байтов по крайнем мере 15 раз). Каждую такую запись байта сопровождать детальным сообщением о действии и числовым значением адреса, по которому размещается этот байт в памяти. Наблюдаемые результаты объяснить. (Базовый вариант) Первую часть задания с двумя процессами, обеспечивающими взаимодействие через разделяемую память разработать для Linux. (Дополнительное задание для максимального рейтинга).

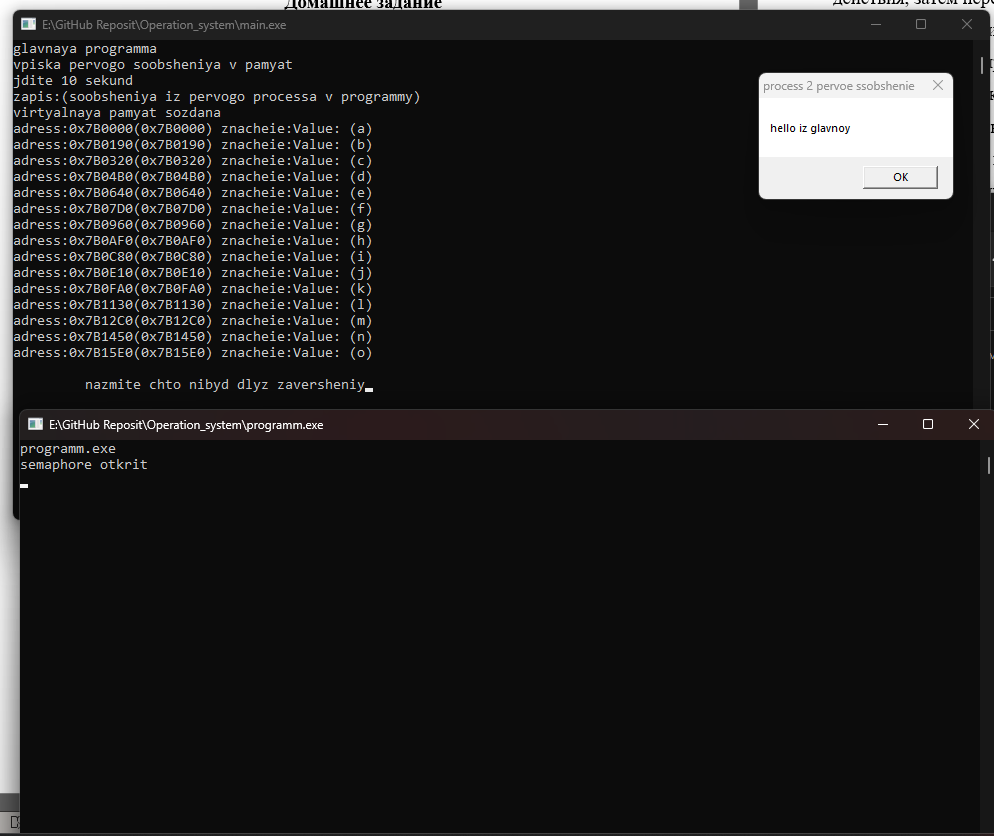
**Листинг программы**

**Programm.c**

#define BUF\_SIZE 15000  
#include <Windows.h>  
#include <stdio.h>  
#include <conio.h>  
#include <tchar.h>  
  
TCHAR szName[] = TEXT("Glavnaya");  
TCHAR fMsg[] = TEXT("hello iz glavnoy");  
TCHAR szMsg[] = TEXT("s pervogo");  
LPCWSTR Semaphore = (LPCWSTR) "Семафор";  
  
char AllocText[] = "abcdefghijklmno";  
  
int main()  
{  
 system("cls");  
 SetConsoleOutputCP(1251);  
 SetConsoleCP(1251);  
 char buff[255];  
 int len;  
 HANDLE Console = GetStdHandle(STD\_ERROR\_HANDLE);  
 len = sprintf(buff, "glavnaya programma\n");  
 WriteFile(Console, buff, len, NULL, NULL);  
 HANDLE hSemaphore = CreateSemaphore(NULL, 0, 1, Semaphore);  
  
 HANDLE hMapFile = CreateFileMapping(  
 INVALID\_HANDLE\_VALUE,  
 NULL,  
 PAGE\_READWRITE,  
 0,  
 BUF\_SIZE,  
 szName);  
 LPTSTR pBuf = (LPTSTR)MapViewOfFile(hMapFile,  
 FILE\_MAP\_ALL\_ACCESS,  
 0,  
 0,  
 0);  
  
 len = sprintf(buff, "vpiska pervogo soobsheniya v pamyat\n");  
 WriteFile(Console, buff, len, NULL, NULL);  
 CopyMemory((PVOID)pBuf, fMsg, (\_tcslen(fMsg) \* sizeof(TCHAR)));  
 Sleep(2000);  
 STARTUPINFO sInfo;  
 PROCESS\_INFORMATION pInfo;  
 ZeroMemory(&sInfo, sizeof(STARTUPINFO));  
 sInfo.cb = sizeof(sInfo);  
 TCHAR path[MAX\_PATH] = \_T("programm.exe");  
 if (CreateProcess(NULL, path, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL,  
 NULL, &sInfo, &pInfo) == 0)  
 {  
 len = sprintf(buff, "file programm.exe ne naiden\n");  
 WriteFile(Console, buff, len, NULL, NULL);  
 }  
 len = sprintf(buff, "jdite 10 sekund\n");  
 WriteFile(Console, buff, len, NULL, NULL);  
 Sleep(10000);  
 len = sprintf(buff, "zapis:");  
 WriteFile(Console, buff, len, NULL, NULL);  
 len = sprintf(buff, "(soobsheniya iz pervogo processa v programmy)\n");  
 WriteFile(Console, buff, len, NULL, NULL);  
 CopyMemory((PVOID)pBuf, szMsg, (\_tcslen(szMsg) \* sizeof(TCHAR)));  
  
 ReleaseSemaphore(hSemaphore, 1, NULL);  
  
 LPTSTR NewMem = (LPTSTR)VirtualAlloc(  
 NULL,  
 BUF\_SIZE,  
 MEM\_COMMIT | MEM\_RESERVE,  
 PAGE\_READWRITE);  
  
 if (NewMem != NULL)  
 {  
 len = sprintf(buff, "virtyalnaya pamyat sozdana\n");  
 WriteFile(Console, buff, len, NULL, NULL);  
 }  
 else  
 {  
 len = sprintf(buff, "ne vishlo\n");  
 WriteFile(Console, buff, len, NULL, NULL);  
 \_getch();  
 return 1;  
 }  
  
 LPTSTR Address = NewMem;  
  
 for (int i = 0; i < 15; i++)  
 {  
 CopyMemory((LPVOID)Address, (PVOID)AllocText, sizeof(char));  
 len = sprintf(buff, "adress:");  
 WriteFile(Console, buff, len, NULL, NULL);  
 len = sprintf(buff, "0x%X", Address);  
 WriteFile(Console, buff, len, NULL, NULL);  
 printf("(0x%X) ", Address);  
 len = sprintf(buff, "znacheie:");  
 WriteFile(Console, buff, len, NULL, NULL);  
 printf("Value: ");  
 len = sprintf(buff, "(%c)\n", AllocText[i]);  
 WriteFile(Console, buff, len, NULL, NULL);  
 Address += 400; // сдвиг  
 Sleep(200);  
 }  
 len = sprintf(buff, "\n\t nazmite chto nibyd dlyz zaversheniy");  
 WriteFile(Console, buff, len, NULL, NULL);  
 getchar();  
 return 0;  
}

**Program.c**

#define BUF\_SIZE 1024  
#include <Windows.h>  
#include <stdio.h>  
#include <conio.h>  
  
TCHAR szName[] = TEXT("Glavnaya");  
TCHAR vaName[] = TEXT("VMAlloc");  
LPCWSTR Semaphore = (LPCWSTR) "Семафор";  
  
int main()  
{  
 SetConsoleOutputCP(1251);  
 SetConsoleCP(1251);  
 char buff[255];  
 int len;  
 HANDLE hMapFile;  
 LPCTSTR pBuf, NewMem;  
 HANDLE hSemaphore = OpenSemaphore(SEMAPHORE\_ALL\_ACCESS, 0,  
 Semaphore);  
 HANDLE Console = GetStdHandle(STD\_ERROR\_HANDLE);  
 len = sprintf(buff, "programm.exe \n");  
 WriteFile(Console, buff, len, NULL, NULL);  
  
 if (hSemaphore != NULL)  
 {  
 len = sprintf(buff, "semaphore otkrit\n");  
 WriteFile(Console, buff, len, NULL, NULL);  
 }  
 else  
 {  
 len = sprintf(buff, "Error: %d\n", GetLastError());  
 WriteFile(Console, buff, len, NULL, NULL);  
 \_getch();  
 return 1;  
 }  
  
 hMapFile = OpenFileMapping(  
 FILE\_MAP\_ALL\_ACCESS,  
 FALSE,  
 szName);  
  
 pBuf = (LPTSTR)MapViewOfFile(hMapFile,  
 FILE\_MAP\_ALL\_ACCESS,  
 0,  
 0,  
 BUF\_SIZE);  
  
 MessageBox(NULL, pBuf, TEXT("process 2 pervoe ssobshenie"), MB\_OK);  
 WaitForSingleObject(hSemaphore, INFINITE);  
 MessageBox(NULL, pBuf, TEXT("process 2 vtoroe"), MB\_OK);  
 len = sprintf(buff, "\n\t najmite");  
 WriteFile(Console, buff, len, NULL, NULL);  
  
 while (1)  
 {  
 if (GetAsyncKeyState(27))  
 {  
 UnmapViewOfFile(pBuf);  
 CloseHandle(hMapFile);  
 CloseHandle(hSemaphore);  
 break;  
 }  
 }  
 return 0;  
}

****

*Рис 1. Пример работы программы*