Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №3

по дисциплине «Основы операционных систем»

на тему «Работа с разделяемой памятью»

Выполнили студенты группы 22ВВП1

Хоссейни Нежад С.А.С.М.

Захаров А. С.

Приняли:

Егоров В.Ю.

Федюнин Р.Н.

Пенза 2024

**Название**

Работа с разделяемой памятью.

**Цель работы**

Изучение способов работы с разделяемой памятью на основе файлов,

проецируемых в память. Изучение способов синхронизации процессов и

нитей с использованием мьютексов (mutex) и семафоров.

**Лабораторное задание**

Разработать систему из двух программ. Первая программа

предназначена для ввода системных команд с командной строки.

Вторая программа демонстрирует результаты выполнения команд.

Синхронизация с использованием мьютексов.

**Описание программы**

В программе Write в начале создаем сразу заблокированный мьютекс, отображение файла и отображаем файл. Запускаем программу Read. Считываем введенную пользователем команду и заносим ее в разделяемую память. В конце программы освобождаем мьютекс.

В программе Read в начале открываем мьютекс и отображение файла, отображаем файл. Ждем освобождение мьютекса, после его захвата выполняем в командной строке считанные данные с разделяемой памяти. В конце освобождаем мьютекс.

**Текст программы**

**Программа Write**

#include <locale.h>

#include <windows.h>

#include <stdio.h>

#include <string.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#define MAX\_DATA\_SIZE 256

#define MUTEX\_NAME "WriteRead"

#define FILE\_MAPPING\_NAME "FileMapping"

void startReadProgram()

{

PROCESS\_INFORMATION pi;

STARTUPINFO si;

WCHAR name[] = L"E:\\OOS\\Lab3\\Read\\x64\\Debug\\Read.exe";

ZeroMemory(&si, sizeof(si));

si.cb = sizeof(si);

ZeroMemory(&pi, sizeof(pi));

if (!CreateProcess(NULL, name, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))

{

printf("CreateProcess failed (%d).\n", GetLastError());

return;

}

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

}

int main() {

setlocale(LC\_ALL, "Rus");

char command[100];

HANDLE hMutex = CreateMutex(NULL, TRUE, MUTEX\_NAME);

if (hMutex == NULL)

{

printf("Не удалось создать мьютекс (%d).\n", GetLastError());

return 1;

}

HANDLE hFileMapping = CreateFileMapping(INVALID\_HANDLE\_VALUE, NULL, PAGE\_READWRITE, 0, MAX\_DATA\_SIZE, FILE\_MAPPING\_NAME);

if (hFileMapping == NULL)

{

printf("Не удалось создать отображение файла (%d).\n", GetLastError());

return 1;

}

LPVOID lpMapAddress = MapViewOfFile(hFileMapping, FILE\_MAP\_ALL\_ACCESS, 0, 0, MAX\_DATA\_SIZE);

if (lpMapAddress == NULL)

{

printf("Не удалось отобразить файл (%d).\n", GetLastError());

CloseHandle(hFileMapping);

CloseHandle(hMutex);

return 1;

}

startReadProgram();

while (1)

{

printf("Программа Write:\n");

printf("Введите команду: ");

fgets(command, sizeof(command), stdin);

command[strcspn(command, "\n")] = '\0';

sprintf\_s((char\*)lpMapAddress, MAX\_DATA\_SIZE, command);

ReleaseMutex(hMutex);

Sleep(100);

WaitForSingleObject(hMutex, INFINITE);

}

UnmapViewOfFile(lpMapAddress);

CloseHandle(hFileMapping);

CloseHandle(hMutex);

return 0;

}

**Программа Read**

#include <locale.h>

#include <windows.h>

#include <stdio.h>

#include <string.h>

#define MAX\_DATA\_SIZE 256

#define MUTEX\_NAME "WriteRead"

#define FILE\_MAPPING\_NAME "FileMapping"

int main()

{

setlocale(LC\_ALL, "Rus");

HANDLE hMutex = OpenMutex(SYNCHRONIZE, FALSE, MUTEX\_NAME);

if (hMutex == NULL)

{

printf("Не удалось открыть мьютекс (%d).\n", GetLastError());

return 1;

}

HANDLE hFileMapping = OpenFileMapping(FILE\_MAP\_ALL\_ACCESS, FALSE, FILE\_MAPPING\_NAME);

if (hFileMapping == NULL)

{

printf("Не удалось открыть отображение файла (%d).\n", GetLastError());

return 1;

}

LPVOID lpMapAddress = MapViewOfFile(hFileMapping, FILE\_MAP\_ALL\_ACCESS, 0, 0, MAX\_DATA\_SIZE);

if (lpMapAddress == NULL)

{

printf("Не удалось отобразить файл (%d).\n", GetLastError());

CloseHandle(hFileMapping);

CloseHandle(hMutex);

return 1;

}

while (1)

{

WaitForSingleObject(hMutex, INFINITE);

printf("\nПрограмма Read:\n");

int result = system((char\*)lpMapAddress);

if (result == -1)

{

printf("Ошибка команды\n");

}

ReleaseMutex(hMutex);

Sleep(200);

}

UnmapViewOfFile(lpMapAddress);

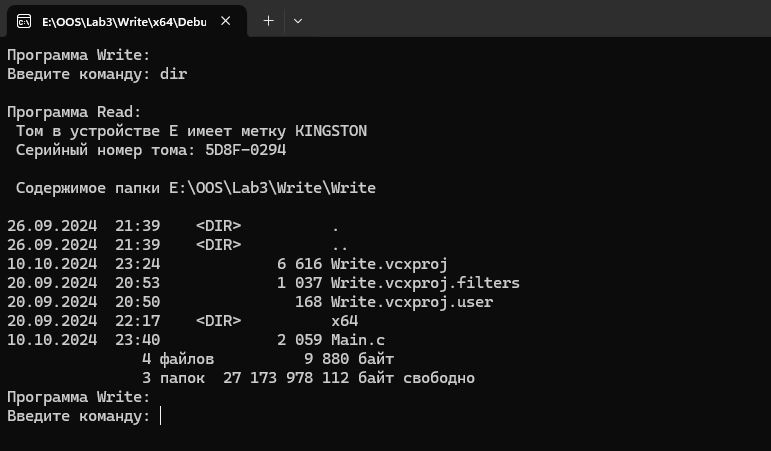
CloseHandle(hFileMapping);

CloseHandle(hMutex);

return 0;

}

**Результат работы программы**

****

**Вывод**

Изучили способы работы с разделяемой памятью на основе файлов,

проецируемых в память. Изучили способы синхронизации процессов и

нитей с использованием мьютексов (mutex) и семафоров.