Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Операционные системы»

III семестр

Задание 2: «Обеспечение обмена данных между процессами посредством технологии File mapping»

|  |  |
| --- | --- |
| Группа: | М8О-108Б-18, №6 |
| Студент: | Васильева Василиса Евгеньевна |
| Преподаватель: | Миронов Евгений Сергеевич |
| Оценка: |  |
| Дата: | 22.03.2020 |

Москва, 2019

1. **Задание**

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 14: Родительский процесс считывает числа со стандартного входного ввода. Дочерний процесс вычисляет квадратный корень этих чисел и передает результаты на печать родительскому процессу.

1. **Адрес репозитория на GitHub**

https://github.com/vasilisavasileva/OS\_3

1. **Код программы на С++**

Client.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#define FILE "Local\\FILE"

#define EV1 "Local\\EV1"

#define EV2 "Local\\EV2"

#include<Windows.h>

#include<tchar.h>

#include<stdbool.h>

int \_tmain(int argc, \_TCHAR\* argv[]) {

SECURITY\_ATTRIBUTES sa;

sa.nLength = sizeof(SECURITY\_ATTRIBUTES);

sa.lpSecurityDescriptor = NULL;

sa.bInheritHandle = TRUE;

HANDLE mp = CreateFileMapping(INVALID\_HANDLE\_VALUE, NULL, PAGE\_EXECUTE\_READWRITE, 0, sizeof(double), FILE);

HANDLE EVENT1 = CreateEvent(NULL, TRUE, TRUE, EV1);

HANDLE EVENT2 = CreateEvent(NULL, TRUE, FALSE, EV2);

TCHAR SecondProcess[] = "OS4\_server";

PROCESS\_INFORMATION ProcessInfo;

ZeroMemory(&ProcessInfo, sizeof(PROCESS\_INFORMATION));

STARTUPINFO StartupInfo;

ZeroMemory(&StartupInfo, sizeof(STARTUPINFO));

StartupInfo.cb = sizeof(STARTUPINFO);

StartupInfo.dwFlags = STARTF\_USESTDHANDLES;

BOOL process = CreateProcess(NULL,

SecondProcess,

NULL, NULL, TRUE,

CREATE\_NEW\_CONSOLE,

NULL, NULL,

&StartupInfo,

&ProcessInfo);

double l;

BOOL isSuccess;

double\* symb = (double\*)MapViewOfFile(mp, FILE\_MAP\_ALL\_ACCESS, 0, 0, sizeof(double));

if (process == 1)

printf("process true\n");

else {

printf("error\n");

exit(1);

}

printf("Enter the count of number: ");

int n;

scanf("%d", &n);

\*symb = n;

ResetEvent(EVENT1);

SetEvent(EVENT2);

WaitForSingleObject(EVENT1, INFINITE);

for (int i = 0;i < n;i++) {

scanf("%lf", &l);

\*symb = l;

ResetEvent(EVENT1);

SetEvent(EVENT2);

WaitForSingleObject(EVENT1, INFINITE);

l = \*symb;

printf("res: %f\n", l);

}

CloseHandle(ProcessInfo.hThread);

CloseHandle(ProcessInfo.hProcess);

CloseHandle(EVENT1);

CloseHandle(EVENT2);

CloseHandle(mp);

system("pause");

return 0;

}

server.cpp

#include<stdio.h>

#include<math.h>

#include<Windows.h>

#include<tchar.h>

#define FILE "Local\\FILE"

#define EV1 "Local\\EV1"

#define EV2 "Local\\EV2"

int \_tmain(int argc, \_TCHAR\* argv[]) {

HANDLE mp = OpenFileMapping(FILE\_MAP\_ALL\_ACCESS, FALSE, FILE);

double\* symb = (double\*)MapViewOfFile(mp, FILE\_MAP\_ALL\_ACCESS, 0, 0, sizeof(double));

HANDLE EV\_1 = OpenEvent(EVENT\_ALL\_ACCESS, FALSE, EV1);

HANDLE EV\_2 = OpenEvent(EVENT\_ALL\_ACCESS, FALSE, EV2);

double l;

int n;

WaitForSingleObject(EV\_2, INFINITE);

n = \*symb;

ResetEvent(EV\_2);

SetEvent(EV\_1);

for (int i = 0;i < n;i++) {

WaitForSingleObject(EV\_2, INFINITE);

l = \*symb;

l = sqrt(l);

\*symb = l;

ResetEvent(EV\_2);

SetEvent(EV\_1);

}

CloseHandle(EV\_1);

CloseHandle(EV\_2);

CloseHandle(mp);

return 0;

}

1. **Результаты выполнения тестов**

|  |  |
| --- | --- |
| Входные данные | Результат |
| 4 | res: 2.000000 |
| 90 | res: 9.486833 |
| 25 | res: 5.000000 |

1. **Объяснение результатов работы программы**

Смысл лабораторной в изучении виртуальной памяти, механизма file mapping и способов взаимодействия с ней. Виртуальная память - это механизм, при котором в работе используется область основной памяти в качестве оперативной. В нашей работе мы используем выделенную область виртуальной памяти для общения между процессами. Мы создаем эту область mapping внутри основного процесса, который реализован в клиенте. Потом создаем второй процесс, а из него открываем доступ к маппингу. В первом процессе считываем количество чисел, которые будет вводить пользователь, принимаем его во втором процессе. Поочередно считываем с потока чиселки, записываем их маппинг, принимаем с другой стороны, вычисляем и закидываем обратно. Для передачи всех данных, которые нам необходимы, делаем явное приведение указателей. Для синхронизации используем event. Первый ивент активен, пока работает клиент, потом вручную приводим второй ивент в сигнальное состояние, а первый усыпляем и переходим к выполнению второго процесса, где все наоборот.

1. **Вывод**

Сам механизм виртуальной памяти интересен тем, что мы можем использовать в качестве оперативной памяти ту память, которая ей по сути не является. В нашем конкретном случае, интересно использовать такую область памяти для общения процессов. Так мы познакомились именно с таким применением этого механизма.