БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий

и робототехники

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

О Т Ч Ё Т

о лабораторной работе № 2

Дисциплина

«Системное программирование»

Тема

«Передача информации между процессами»

Выполнил: студент гр. 10702217 Храмков Д. С.

Проверил: Разорёнов Н. А.

Минск 2019

***Лабораторная работа № 2***

**ПЕРЕДАЧА ИНФОРМАЦИИ МЕЖДУ ПРОЦЕССАМИ**

**Цель работы:** изучить вопросы проектирования и создания модальных и немодальных диалоговых панелей на базе Win32/64 API.

**Изучаемые вопросы**

1. Функции для работы с файлами, отображаемыми в память;
2. Файлы данных, проецируемые в память;
3. Совместный доступ процессов к данным через механизм проецирования;
4. Обработка текстового и графического файлов двумя процессами;
5. Работа с файлами больших размеров;
6. Взаимодействие процессов через Page файл.

***Постановка задачи***

1. Осуществлять обмен текстовыми и графическими данными между двумя приложениями P1 и P2 с выводом информации в заданную область окна программы;
2. Реализовать обработку процессом файлов больших размеров;
3. Осуществить обмен данными меджу двумя приложениями P1 и P2 через Page файл с выводом информации в заданную область окна программы
4. **Функции для работы с файлами, отображаемыми в память.**

Для создания проекции в память файла используется функция CreateFileMapping. Для того, чтобы получить эту проекцию другим процессом используют OpenFileMapping. В примере ниже получение проекции осуществляется через уникальное имя проекции, присваиваемое ей на этапе создания.

hMappingFileTxt = CreateFileMapping(hFileTxt, NULL, PAGE\_READWRITE, 0, txt\_size, this\_txt\_mapping\_name);

hMappingFileBmp = CreateFileMapping(hFileBmp, NULL, PAGE\_READONLY, 0, 0, this\_bmp\_mapping\_name);

hAnotherMappingFileTxt = OpenFileMapping(FILE\_MAP\_READ, FALSE, txt);

hAnotherMappingFileBmp = OpenFileMapping(FILE\_MAP\_READ, FALSE, bmp);

1. **Файлы данных, проецируемые в память.**

Для работы с отображаемыми в память файлами нужно:

* Создать или открыть файл;
* Создать объект ядра проекция файла, чтобы сообщить системе размер файла и способ доступа к нему;
* Указать системе, как спроецировать в адресное пространство объект проекция файла – целиком или частично.

По завершении работы с проецируемым файлом следует:

* Сообщить системе об отмене проецирования на адресное пространство процесса объекта ядра проекция файла;
* Закрыть этот объект;
* Закрыть объект ядра файл.

hAnotherProjectionFileTxt = MapViewOfFile(hAnotherMappingFileTxt, FILE\_MAP\_READ, 0, 0, 0);

hAnotherProjectionFileBmp = MapViewOfFile(hAnotherMappingFileBmp, FILE\_MAP\_READ, 0, 0, 0);

UnmapViewOfFile(hAnotherProjectionFileTxt);

UnmapViewOfFile(hAnotherProjectionFileBmp);

CloseHandle(hAnotherMappingFileTxt);

CloseHandle(hAnotherMappingFileBmp);

1. **Совместный доступ процессов к данным через механизм проецирования.**

Совместный доступ процессов к данным через механизм проецирования позволяет наблюдать за изменениями в проецируемом файле в режиме реального времени, т.е. если первый процесс каким-либо образом изменит проецируемый файл, то второй процесс сможет получить эти изменения сразу же после их применения.

hFileTxt = CreateFile(this\_proc\_txt, GENERIC\_READ | GENERIC\_WRITE, FILE\_SHARE\_READ, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

WriteFile(hFileTxt, mesage1.c\_str(), mesage1.length(), 0, NULL);

hMappingFileTxt = CreateFileMapping(hFileTxt, NULL, PAGE\_READWRITE, 0, txt\_size, this\_txt\_mapping\_name);

hProjectionFileTxt = MapViewOfFile(hMappingFileTxt, FILE\_MAP\_READ, 0, 0, 0);

1. **Обработка текстового и графического файлов двумя процессами.**

Одним из заданий данной лабораторной работы является обработка текстового и графического файлов двумя процессами. Для реализации задачи создавались объекты ядра файл, затем проекция этих файлов, которая впоследствии отображалась на адресное пространство.

hAnotherMappingFileTxt = OpenFileMapping(FILE\_MAP\_READ, FALSE, txt);

hAnotherProjectionFileTxt = MapViewOfFile(hAnotherMappingFileTxt, FILE\_MAP\_READ, 0, 0, 0);

1. **Работа с файлами больших размеров.**

При обработке файлов больших размеров нужно проецировать не весь файл, а его часть. Проецирование стоит начинать с самого начала файла, отправляя в память последовательные кусочки файла. Повторять данное действие надо пока все кусочки не будут отображены в памяти.

while (dwFileSize > 0)

{

DWORD dwBytesInBlock = sinf.dwAllocationGranularity;

if (dwFileSize < sinf.dwAllocationGranularity) {

dwBytesInBlock = (DWORD)dwFileSize;

}

PBYTE hHugeMapFileStartAddr = (PBYTE)MapViewOfFile(hHugeFileMapping, FILE\_MAP\_READ,

(DWORD)(dwFileOffset >> 32), (DWORD)(dwFileOffset & 0xFFFFFFFF), dwBytesInBlock);

UnmapViewOfFile(hHugeMapFileStartAddr);

dwFileOffset += dwBytesInBlock;

dwFileSize -= dwBytesInBlock;

}

1. **Взаимодействие процессов через Page файл.**

Система виртуальной памяти в Win32 использует файл размещения (Page файл), имея возможность преобразования стрнаниц оперативной памяти в страницы файла на диске и наоборот. Система может проецировать на оперативную память не только файл размещения, но и любой другой файл. Приложения могут использовать эту возможность. Это может использоваться для обеспечения более быстрого доступа к файлам, а также для совместного использования памяти.

hMappingFileTxt = CreateFileMapping(INVALID\_HANDLE\_VALUE, NULL, PAGE\_READWRITE, 0, 100, this\_proc\_txt);

if (hMappingFileTxt == NULL) {

MessageBox(NULL, L"File mapping for txt file wasn't created", this\_proc\_txt, MB\_OK | MB\_ICONWARNING);

return -1;

}

else {

hProjectionFileTxt = MapViewOfFile(hMappingFileTxt, FILE\_MAP\_READ, 0, 0, 0);

}

**Вывод:**

* 1. Научились создавать объект ядра проекция;
  2. Научились отображать проекцию файла в память;
  3. Научились осуществлять совместный доступ процессов к данным через проецирование;
  4. Научились работать с файлами больших размеров.