Файловые системы

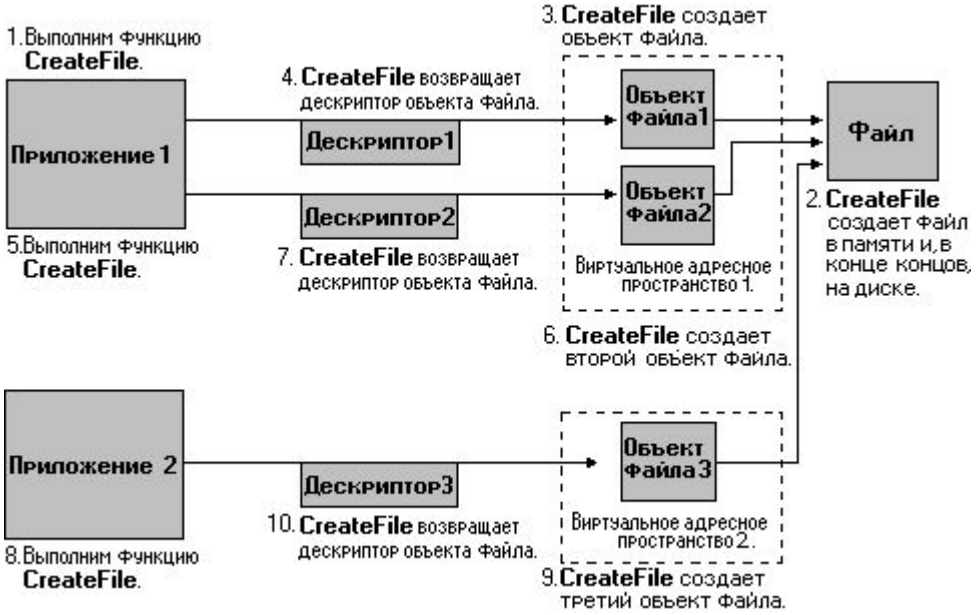
Работа с файлами в Windows API

Работа с файлами в Windows API

Совместная работа с файлами

Совместный доступ к файлу

- Управление объектами типа «файл» отличается от управления другими объектам ядра.
- Вы не можете открыть ранее созданный объект типа «файл», Вы можете только создать новый объект типа «файл» и далее выполнять совместный доступ к файлу на диске.



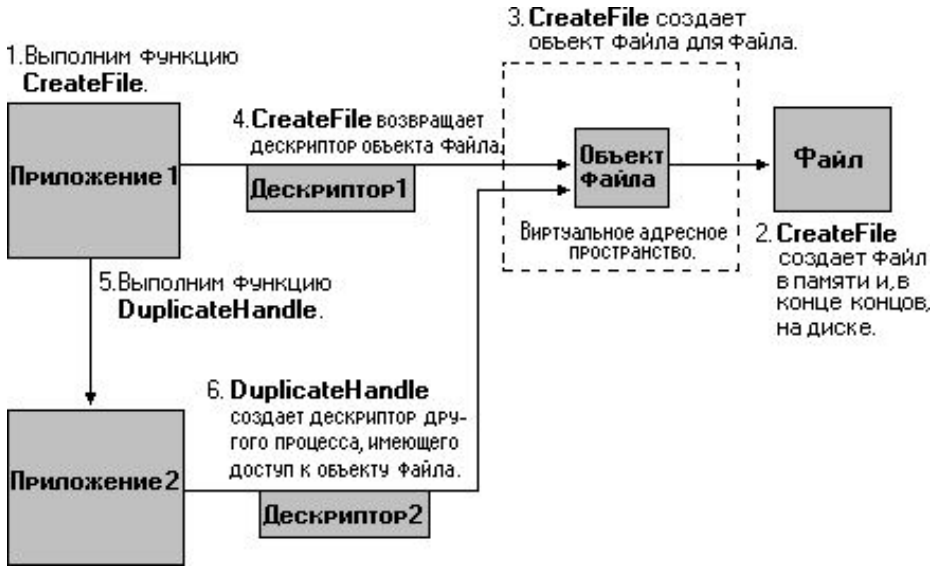
Функция повторного открытия файла

```
HANDLE ReOpenFile(
    HANDLE hOriginalFile, // дескриптор уже открытого файла
    DWORD dwDesiredAccess,  //  права доступа
    DWORD dwShareMode,  // режим совместного доступа
    DWORD dwFlags // флаги
);
```

- **Примечание:** Параметр *dwFlags* не может содержать какой-либо из атрибутов файла (**FILE_ATTRIBUTE_** *). Эти атрибуты могут задаваться только тогда, когда файл создается.

Совместное использование объекта типа «файл»

- Только используя механизм дублирования дескрипторов объектов, Вы можете получить более одного дескриптора для одного и того же объекта типа «файл».



Совместный доступ и блокировка файлов

- Если функции *CreateFile ()* указать режимы совместного использования файла FILE_SHARE_READ или FILE_SHARE_WRITE, то несколько процессов смогут одновременно открыть файлы и выполнять операции чтения и записи. Если же эти режимы не указаны, совместное использование файлов будет невозможно – первый процесс, который откроет файл, заблокирует возможность работы с этим файлом для других процессов.
- Для организации совместного доступа нескольких процессов к разным участкам файла участку файла требуется использовать пару функций *LockFile () / UnlockFile ()* или *LockFileEx () / UnlockFileEx ()*.
- Блокирование файлов является ограниченной разновидностью синхронизации параллельно выполняющихся процессов и потоков, обсуждение вопросов синхронизации будет рассмотрено в соответствующей лекции.

Функции блокировки

- Блокировка участка файла для монопольного доступа выполняется функцией *LockFile ()*, после использования заблокированного участка, а также перед завершением своей работы процессы должны разблокировать все заблокированные ранее участки, вызвав для этого функцию *UnlockFile ()*.
- Функции *LockFileEx ()* и *UnlockFileEx ()* позволяют блокировать/разблокировать участок открытого файла либо для разделяемого доступа (разрешающего доступ одновременно нескольким приложениям в режиме чтения), либо для монопольного доступа (разрешающего доступ только одному приложению в режиме чтения/записи).

Функции *LockFile* и *UnlockFile*

```
BOOL LockFile(
    HANDLE hFile, // дескриптор файла
    DWORD  dwFileOffsetLow,  // младшее слово смещения участка
    DWORD  dwFileOffsetHigh, // старшее слово смещения участка
    DWORD  nNumberOfBytesToLockLow, // младшее слово длины участка
    DWORD  nNumberOfBytesToLockHigh // старшее слово длины участка
);

BOOL UnlockFile(
    HANDLE hFile, //  дескриптор  файла
    DWORD  dwFileOffsetLow,  // младшее слово смещения участка
    DWORD  dwFileOffsetHigh, // старшее слово смещения участка
    DWORD  nNumberOfBytesToUnlockLow, // млад. слово длины участка
    DWORD  nNumberOfBytesToUnlockHigh // старш. слово длины участка
);
```

Функции *LockFileEx* и *UnlockFileEx*

```
BOOL LockFileEx(
    HANDLE hFile, //  дескриптор  файла
    DWORD dwFlags, // вид блокировки и режим ожидания
    DWORD dwReserved, // зарезервировано
    DWORD nNumberOfBytesToLockLow, // млад. слово длины участка
    DWORD nNumberOfBytesToLockHigh, // старш. слово длины участка
    LPOVERLAPPED lpOverlapped // указание начала участка
) ;

BOOL UnlockFileEx(
    HANDLE hFile, //  дескриптор  файла
    DWORD dwFlags, // вид блокировки и режим ожидания
    DWORD dwReserved, // зарезервировано
    DWORD nNumberOfBytesToUnlockLow, // млад. слово длины участка
    DWORD nNumberOfBytesToUnlockHigh, // старш. слово длины участка
    LPOVERLAPPED lpOverlapped // указание начала участка
) ;
```

Параметры *LockFileEx* и *UnlockFileEx*

- *dwFlags* – определяет вид блокировки файла, а также режим ожидания доступности затребованной блокировки:
  - LOCKFILE_EXCLUSIVE_LOCK – запрос монопольной блокировки в режиме чтения/записи. Если это значение не задано, запрашивается разделяемая блокировка (только чтение).
  - LOCKFILE_FAIL_IMMEDIATELY – задает режим немедленного возврата функции с возвращаемым значением равным FALSE, если приобрести блокировку не удается. Если это значение не задано, функция переходит в режим ожидания.
- *lpOverlapped* – используется для указания 64-битовой позиции начала участка файла, подлежащего блокированию;
- *dwReserved* – значение этого параметра должно быть равным 0.

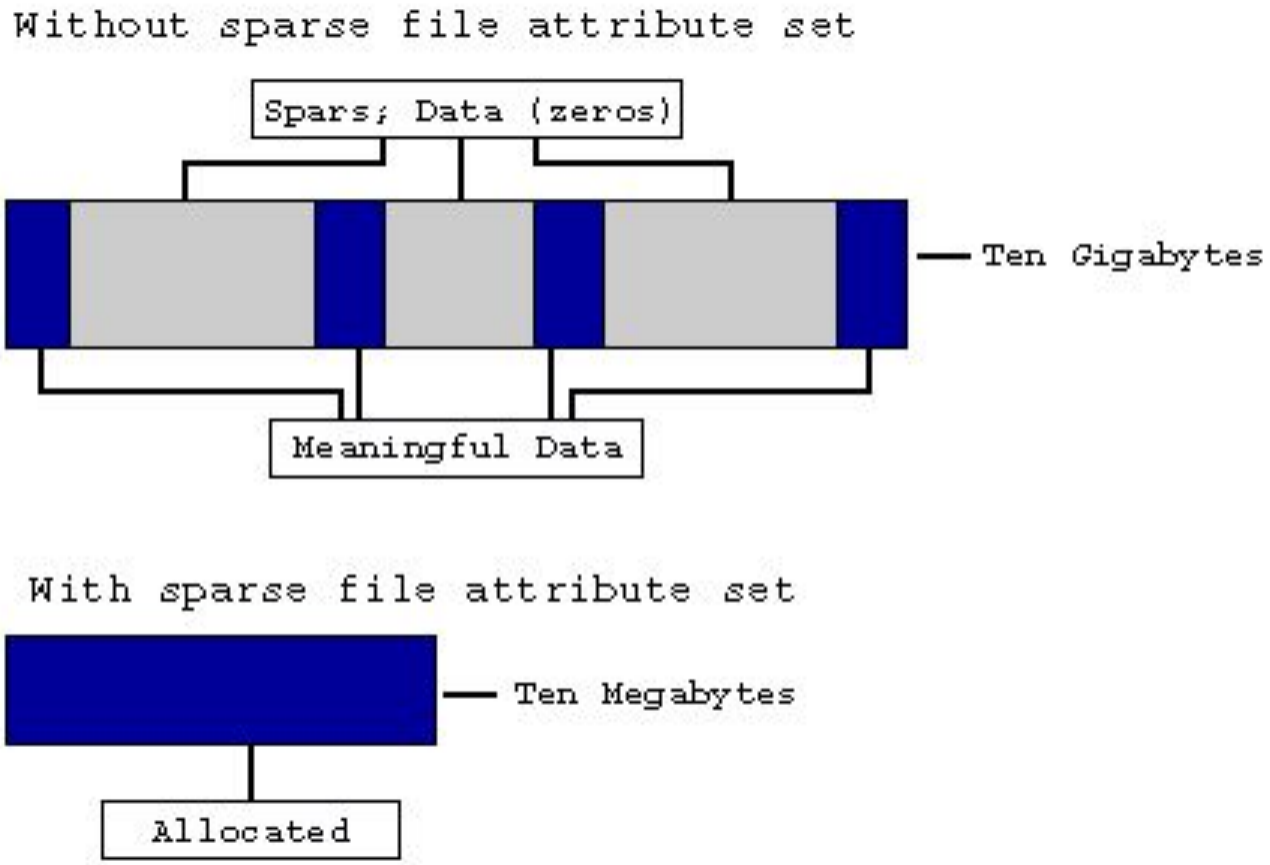
Особенности блокирования участков файла

- Вновь создаваемая и существующие области блокирования в файле не могут перекрываться.
- Возможно блокирование участка, границы которого выходят за пределы файла. Такая операция может оказаться полезной в случае расширения файла процессом или потоком.
- Блокировки не наследуются вновь создаваемыми процессами.
- Границы участка разблокирования должны в точности совпадать с границами ранее заблокированной области. Любая попытка разблокирования участка, не совпадающего в точности с одной из существующих заблокированных областей, будет неудачной (функция вернет FALSE).

Работа с файлами в Windows API

Работа с «разреженными» файлами

Разреженные файлы (sparse files)



Создание разреженных файлов

- Прежде чем создать разреженный файл необходимо проверить поддержку **FILE_SUPPORTS_SPARSE_FILES** со стороны файловой системы.
- Для создания разреженного файла необходимо установить этому файлу атрибут **FILE_ATTRIBUTE_SPARSE_FILE**.
- С помощью функции *CreateFile ()* установить атрибут разреженного файла невозможно, для установки атрибута разреженного файла необходимо воспользоваться специальной функцией управления устройствами ввода-вывода *DeviceIoControl ()* с управляющим кодом **FSCTL_SET_SPARSE.**
- Единственный способ сбросить бит этого атрибута состоит в том, чтобы переписать файл, например, при помощи вызова функция *CreateFile ()* с флагом **CREATE_ALWAYS**.

Пример проверки поддержки разреженных файлов

```
char   szVolName[MAX_PATH],  szFSName[MAX_PATH];
DWORD  dwSN, dwMaxLen, dwVolFlags;

GetVolumeInformation("C:\\", szVolName, MAX_PATH, &dwSN,
      &dwMaxLen, &dwVolFlags, szFSName, MAX_PATH);

if (dwVolFlags & FILE_SUPPORTS_SPARSE_FILES)
{
      // File system supports sparse streams
} else {
      // Sparse streams are not supported
}
```

Функция *DeviceIoControl*

```
BOOL DeviceIoControl (
   HANDLE hDevice, // дескриптор устройства
   DWORD dwIoControlCode, // код операции
   LPVOID lpInBuffer, // буфер входных данных
   DWORD nInBufferSize, // размер буфера входных данных
   LPVOID lpOutBuffer, // буфер данных результата
   DWORD nOutBufferSize, // размер буфера результата
   LPDWORD lpBytesReturned, // адрес данных для вывода
   LPOVERLAPPED lpOverlapped  // адрес структуры
                  // OVERLAPPED
);
```

Пример создания разреженного файла

```
HANDLE hFile = CreateFile("C:\\Sparse.dat", GENERIC_WRITE,  0,
NULL, CREATE_NEW,  FILE_ATTRIBUTE_NORMAL, NULL);

DWORD dwTemp;

DeviceIoControl(hFile, FSCTL_SET_SPARSE, NULL, 0, NULL, 0,
&dwTemp, NULL);}
```

Особенности вызова *DeviceIoControl*

- Если параметр *hDevice* открывался без установки флажка FILE_FLAG_OVERLAPPED, параметр *lpOverlapped* игнорируется.
- Если параметр *hDevice* открывался с флагом FILE_FLAG_OVERLAPPED, операция выполняется как перекрывающая (асинхронная). В этом случае, параметр *lpOverlapped* должен указать на допустимую структуру OVERLAPPED, которая содержит дескриптор объекта события. Иначе, функция завершается ошибкой непредсказуемыми способами.
- Если параметр *lpOverlapped* – NULL, то *lpBytesReturned* не может быть NULL.

Задание разреженных областей файла

- Для задания в разреженном файле «нулевых» областей необходимо использовать функцию *DeviceIoControl ()* с управляющим кодом **FSCTL_SET_ZERO_DATA** .
- Начало и конец создаваемой «нулевой» области задаются через параметр *lpInBuffer* функции *DeviceIoControl ()*.
- Если используете функцию *DeviceIoControl ()* с кодом **FSCTL_SET_ZERO_DATA** в отношении неразреженного файла, то в файл будут записаны «физические» нули.

Пример создания разреженной области файла

```
FILE_ZERO_DATA_INFORMATION fzdi;
DWORD dwTemp;

fzdi.FileOffset.QuadPart = uAddress; fzdi.BeyondFinalZero.QuadPart =
      uAddress + uSize;

DeviceIoControl(hFile, FSCTL_SET_ZERO_DATA, &fzdi, sizeof(fzdi),
NULL, 0, &dwTemp, NULL);
```

Задание разреженной области в конце файла

- Для задания разреженной области в конце файла нет необходимо использовать функцию *DeviceIoControl ()* с управляющим кодом **FSCTL_SET_ZERO_DATA** .
- Достаточно установить указатель файла в новую позицию за «пределами» файла и затем вызвать функцию *SetEndOfFile ()*.

Получение информации о разреженных областях файла

- Для получения информации о разреженных областях файла необходимо использовать функцию *DeviceIoControl ()* с управляющим кодом **FSCTL_QUERY_ALLOCATED_RANGES**.
- Пример кода программы, определяющей расположение разреженных областей файла приведен в статье http://www.flexhex.com/docs/articles/sparse-files.phtml