Весна 2021

Системное программное обеспечение

Онлайн-лекции

Лекция №10: Файлы и работа с файлами

Доцент, к.т.н. ГОЛЬЦОВ Александр Геннадьевич

Файлы

- Файлы - это именованные порции данных произвольного размера в долговременной памяти (на диске, магнитной ленте, flash-накопителе).
- Доступ к файлам для программ организуется средствами операционной системы.
- Поддержка файловой системы - функция ОС, "делающая ее Операционной Системой". Согласно негласному правилу, различные комплексы управляющих программ получают право называться ОС, если поддерживают файловую систему.

Файловая система

- ФС - это набор соглашений, действующих в рамках ОС и определяющих способ хранения, именования и доступа к файлам на носителе.
- ФС в ДОС подразумевает возможность объединения файлов в каталоги (директории), в предшественнице ДОС - CP/M - директорий не было, на носителе (дискете) был просто набор файлов.
- Каждому физическому или логическому диску соответствует том (volume); в каждом томе существует корневая директория и вложенные в нее директории.
- Директория = каталог = папка (папки в Windows)

Обычные и служебные файлы

- Обычные файлы имеют имя и содержат данные, которые используются различными программами.
- Служебные файлы являются элементами файловой системы и непосредственно обеспечивают ее функционирование.
- Служебные файлы ДОС:
  - директории=каталоги (кроме корневой!)
  - метки тома
- В других ОС дополнительно: история транзакций с файлами, загрузочный сектор, карта свободного места тома и т.п.

Имена файлов

- Имена файлов и директорий подчиняются правилу 8+3: до 8 символов в имени файла и до 3 символов в расширении.
- Имена файлов при работе с функциями ДОС записываются или в формате "name8.ext", или под имя и расширение отводятся массивы из 8 и 3 символов соответственно, "хвост" имени и расширения в этом случае заполняется пробелами.
- Максимальная длина пути к файлу - 80 символов.

Путь (path)

- Путь является полным адресом файла в компьютере.
- На все же это ПУТЬ, а не "адрес файла"!
- Структура пути: C:\LABS\A-08\LAB3.ASM
  C: - буква диска, может отсутствовать, \ - ссылка на корневой каталог, LABS\, A-08\ - ссылки на подкаталоги, LAB3.ASM - имя файла
- Если не указать диск - отсчет ведется с текущего диска, если не указать корневую директорию (ведущий бэкслэш) - отсчет ведется от текущей директории

Текущая и домашняя директории

- Домашняя директория программы (home dir) - откуда программа запущена, где лежит ее исполнимый файл.
- Текущая директория (current dir) - директория "по умолчанию", на каждом диске - своя, устанавливается и может меняться в процессе работы программы.
- Текущий диск - тоже может меняться в ходе работы программы.
- При старте программы с предварительным переходом в ее директорию, как правило, текущие диск и директория соответствуют домашней.
- А так - нет (указан полный путь): c:\labs\lab3.exe

Командная строка и управление директориями

- **буква:**, например, **d:** - установить текущий диск (никакие директории не упоминаются!)
- **cd путь** - установить текущую директорию
  - cd lab1 - в текущей директории найти lab1 и сделать ее текущей
  - cd \lab1 - сделать текущей директорию lab1, расположенную в корневом каталоге текущего диска
  - cd d:\lab1 - на диске d: сделать текущей директорию \lab1 в корне, но это не приводит к смене текущего диска

Именование файлов

- В именах файлов допустимы:
  - буквы латинского алфавита
  - цифры
  - подчеркивания, ~
- Недопустимы: ?, *, запятые, кавычки, пробелы, точки
- Буквы русского или национального алфавита
- Вообще - не называйте файлы и папки в проекте длинными и русскими именами, даже если это разрешено ОС! Русские имена - для документов пользователя.

Длинные имена файлов

- В Windows NT / Windows 95 появляются длинные имена файлов, где допустимы пробелы, точки и прочие вольности.
- Программы DOS, запускаемые в этих ОС, могли работать с такими файлами, используя псевдонимы: имена в формате 8.3, построенные по определенным правилам:

"c:\program files\my dir\file name.txt"

➔

C:\PROGRA~1\MYDIR~1\FILENA~1.TXT

Чтение и запись

- Файлы подразумевают последовательный доступ.
- В файл можно писать/читать произвольное количество байт за одно обращение.
- Следующая порция записываемой информации помещается за текущей (считываемой - считывается после текущей).
- Текущая позиция - смещение относительно начала файла в байтах. Это логический указатель, связанный с каждым файлом и изменяемый при чтении-записи или явно.
- Текущая позиция одна и для чтения, и для записи.
- При чтении/записи ТП смещается к концу файла на размер прочитанного/записанного блока данных.
- Текущую позицию можно явно менять, организуя таким образом прямой доступ к произвольной части файла.

Чтение и запись

- Возможны с одним и тем же файлом в произвольной последовательности во времени.
- Читаем/пишем, начиная с текущей позиции.
- Если в ходе записи выходим за конец файла - файл увеличивается.
- Если в ходе чтения выходим за конец файла - считываем меньшее количество байт, чем заказывали.
- В ЯВУ аналогичные функции зовутся BlockRead / BlockWrite (Паскаль), ReadFile / WriteFile (WinAPI), fread / fwrite (Си).
- Поверх файлов удобно организовывать потоки данных (stream).

Один байт и целый блок

- Операции чтения/записи в файлы физически выполняются секторами - блоками данных 512 байт.
- Если нужно прочитать один байт, то читается весь сектор, и из него копируется нужная порция данных в качестве результата.
- Если нужно записать один байт, то читается весь сектор, модифицируется и записывается целиком.
- ➔ Читать и писать в файлы побайтно можно, но очень неэффективно, если речь идет про большие порции данных. Идет потеря производительности в сотни раз.

Свойства файла

- Имя и расширение (8+3)
- Путь (до 80 символов с именем файла, до 64 символов - только каталоги)
- Размер (до 2 Гб)
- Дата изменения
- Атрибуты (набор битов):
  - архивный
  - только чтение
  - системный
  - скрытый
  - директория
  - метка тома

"Бинарные", "текстовые" и прочие файлы

- Любой файл может рассматриваться как бинарный, т.е. состоящий из байтов информации, которые можно читать-писать в произвольном месте в произвольном количестве.
- Могут существовать дополнительные договоренности:
  - о структуре файла (какие байты в каком месте имеют какое значение)
  - о допустимых значениях байтов ("только буквы, цифры и знаки препинания")
  - о том, какое расширение должно быть у файла, хранящего определенную информацию
- Все эти договоренности - "более высокого уровня", они определяют способ работы с конкретными данными, для хранения которых используется файл, но не механизм доступа к самому файлу.

Текстовые файлы

- Содержат буквы, цифры и прочие знаки - байты с кодами в диапазоне 32 (пробел)..255.
- Признак конца строки - пара байт CR-LF (коды 13-10, 0D-0Ah)
- Не предназначены (неудобны) для попеременного чтения и записи в один и тот же файл.
- Байты с кодами <32 считаются управляющими. Изначально управляли текстовыми принтерами - печатными машинками.
  13 (CR) - возврат каретки, перейти к началу строки
  10 (LF) - перевод строки, на строку вниз
  9 (Tab) - табуляция, сдвинуть курсор вправо на ближайшую следующую позицию табуляции (кратно 8 или 4)
  12 (PG) - следующая страницы, прокрутить лист вниз на множество строк до начала следующей страницы
  7 (Bell) - издать звуковой сигнал
  8 (BS) - Back Space, забой
  27 (Esc) - начало управляющей escape-последовательности
- Символ 26 (EOF, конец файла) служил для контроля передачи данных по линии как высокоуровневый маркер конца данных, не требуется, не нужен, не ставится в файле.

Принцип организации информации на диске

- На уровне контроллера диска (аппаратно) существуют дорожки и сектора (раньше они были реальными, сейчас - виртуальные).
- ОС выделяет пространство на диске кластерами, 1 кластер = несколько секторов, например 4К = 8 * 512
- Каждый кластер свободен или принадлежит какому-то файлу (размер занимаемого файлом места на диске округляется в большую сторону до кратного размеру кластера).
- Файл = цепочка кластеров, информация о цепочках (ссылка на следующий кластер) также хранится в служебной области на диске.
- Каталог файлов (директория) - это служебный файл, содержащий имена файлов, даты создания, размеры, атрибуты - и номера первых кластеров в цепочках кластеров, представляющих эти файлы.

Файловая система FAT

Содержимое диска:

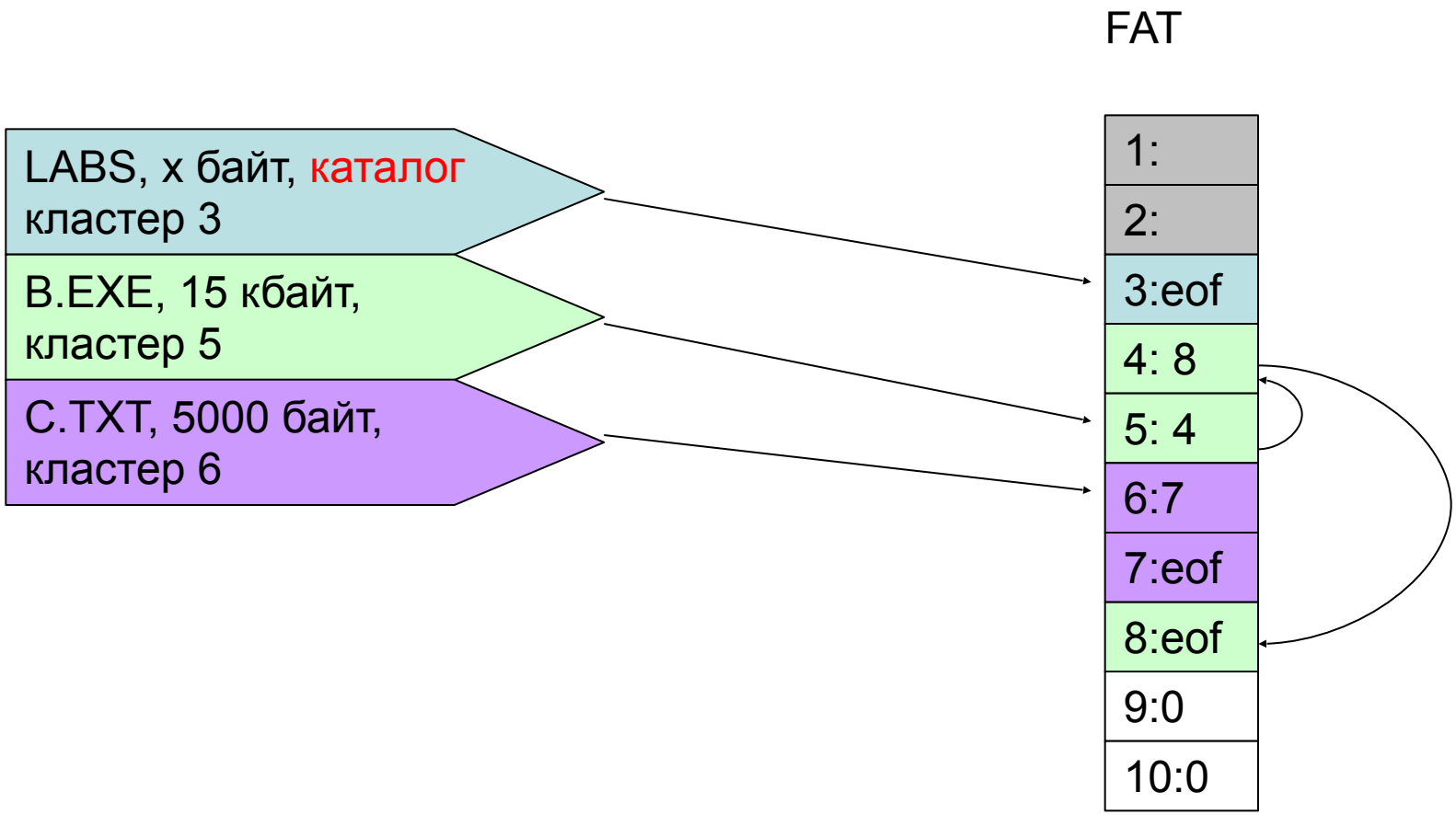
Загрузочный сектор	512 байт
Информация файловой системы	512 байт
Резервные сектора	
Таблица размещения файлов 1 (FAT1)	Число кластеров * 4
Таблица размещения файлов 2 (FAT2)	Число кластеров * 4
Корневой каталог	1 кластер
Массив данных	остальные кластеры

Каждый элемент FAT - номер следующего кластера или признак "конец цепочки" или ноль.

Корневой каталог - занимает 1 кластер (не файл, не имеет имени).

Другие каталоги являются файлами.

Пример: каталог из 3 файлов



Работа с файлами: FCB и хэндлы

- FCB (File Control Block) - низкоуровневая структура (37 байт), поддерживаемая "базовой ДОС" (функциями ОС низкого уровня).
- Это наследие CP/M, неудобно, громоздко.
- Ориентировано на работу с файлами текущего каталога (в CP/M разбиения на иерархические каталоги не было).

- Функции ДОС высокого уровня работают с хэндлами файлов.
- Хэндл - числовой идентификатор файла, назначается ему при открытии.
- Одновременно можно открыть до 16 файлов, включая IN, OUT, AUX и ERR.
- Фраза FILES=nn в файле config.sys позволяет увеличить это число до 255.

Структура FCB

Смещение	Размер	Назначение
0	1	***Диск***: 00 = текущий, 01 = **A:**, 02 = **B:** и т.д.
1 - 8	**8**	***Имя файла***, дополняется пробелами до 8 символов.
9 - 11	**3**	***Расширение файла***, дополняется пробелами до 3 символов.
12 - 13	**2**	***Номер текущего блока***. Блок содержит 128 записей. Для обращения к конкретной записи используется номер текущего блока и номер текущей записи (байт 32). Первый блок файла имеет номер 0, второй - 1 и т.д. Операция открытия файла устанавливает в данном поле 0.
14 - 15	**2**	***Логический размер записи***. Операция открытия устанавливает в этом поле значение 128 (80H). После открытия и перед любой операцией чтения или записи можно устанавливать в данном поле любое требуемое значение длины записи.
16 - 19	**4**	***Размер файла***. При создании файла **DOS** вычисляет и записывает это значение (произведение числа записей на размер записи) в оглавление. Операция открытия выбирает размер файла из оглавления и заносит его в данное поле. Программа может читать значение из этого поля, но не может изменять его.
20 - 21	**2**	***Дата***. При создании или последней модификации файла **DOS** записывает дату в оглавление. Операция открытия выбирает дату из оглавления и заносит ее в данное поле.
22 - 31	**10**	***Зарезервировано***.
32	**1**	***Текущий номер записи***. Данное поле содержит текущий номер записи (0-127) в текущем блоке. **DOS** использует текущие значения блока и записи для локализации записи в дисковом файле. Обычно номер начальной записи в данном поле - 0, но его можно заменить на любое значение от 0 до 127.
33 - 36	**4**	***Относительный номер записи***. Используется для **произвольного доступа** к записи при операциях чтения или записи. Данное поле должно содержать относительный номер записи. В случае произвольного доступа DOS автоматически преобразует относительный номер записи в текущие номера блока и записи.

Доступ через FCB

- Подробно рассматривать не будем.
- Нужно постоянно отслеживать и менять информацию в структуре FCB, заполненной при открытии файла.
- Дополнительно используется буфер DTA (disk transfer area), регистрируемый программой для операций с файлами.
- Функции INT 21 с номерами (задаются в регистре AH):
  - 0Fh - open file
  - 10h - close file
  - 11h - find file
  - 21h - read record
  - 22h - write record
  - 23h - get file size
  - 24h - set position
  - 27h - block read
  - 28h - block write
  - и др., искать по ключевому слову "FCB".

Доступ через хэндлы

- Примерно соответствует логике организации функций работы с файлами в ЯВУ.
- Переменная файлового типа в ЯВУ является записью, одно из полей - хэндл файла.
- Функции:
  - 3Ch - создать файл (перезаписать пустым)
  - 3Dh - открыть существующий файл (для чтения и записи!)
  - 3Eh - закрыть файл
  - 3Fh - прочесть блок данных из файла
  - 40h - записать блок данных в файл
  - 42h - установить явно текущую позицию чтения-записи
- Ключевое слово для многих подобных функций - "DOS 2+"

Атрибуты файла

-	-	arc	dir	vol	sys	hid	ro

Биты байта атрибутов:

0: ro - только чтение
1: hid - скрытый
2: sys - системный
3: vol - метка тома
4: dir - подкаталог
5: arc - архивный

Обычный файл имеет атрибут 0 или 20h (arc)

Основные коды ошибок

- 02 - файл не найден
- 03 - путь не найден
- 04 - слишком много открытых файлов
- 05 - доступ запрещен
- 06 - неверный хэндл
- 16 - попытка удалить каталог как файл
- 18 - нет больше файлов (по условию поиска)

Создать файл (существующий стирается)

INT 21 - DOS 2+ - CREATE A FILE WITH HANDLE (CREAT)

AH = 3Ch
CX = attributes for file
DS:DX = address of ASCIZ filename

Return: CF = 1 if error
AX = Error Code
CF = 0 successful
AX = file handle

```
fname1 db 'my.txt',0
handle1 dw ?
.....................................
mov ah, 3Ch
mov cx,0
mov dx, offset fname1
int 21h
jc ErrorCreate
mov handle1, ax
.....................................
```

Открыть файл

```
INT 21 - DOS 2+         - OPEN DISK FILE WITH HANDLE
      AH = 3Dh
      AL = access code
         0 =  Read Only
         1 =  Write Only
         2 =  Read/Write
      AL bits 7-3 = file-sharing modes (DOS 3.x)
         bit   7   = inheritance flag, set for no inheritance
         bits 4-6 = sharing mode
                    000 compatibility    mode
                    001 exclusive (deny all)
                    010 write                 access denied (deny write)
                    011 read access denied (deny read)
                    100 full access permitted         (deny none)
         bit   3   = reserved, should be zero
      DS:DX =              address      of ASCIZ filename
Return:      CF = 1 if error
         AX = Error Code
      CF = 0 successful
         AX = file handle
```

```
fname1 db 'my.txt',0
handle1 dw ?
.....................................
mov ah, 3Dh
mov al,2 ; r&w
mov dx, offset fname1
int 21h
jc ErrorOpen
mov handle1, ax
.............................
```

Закрыть файл

INT 21 - DOS 2+ - CLOSE A FILE WITH HANDLE
AH = 3Eh
BX = file handle
Return: CF = 1 if error
AX = Error Code

Читать из файла

INT 21 - DOS 2+ - READ FROM FILE WITH HANDLE

AH = 3Fh
BX = file handle
CX = number of bytes to read
DS:DX = address of buffer

Return: CF = 1 if error
AX = Error Code
CF = 0 successful
AX = number of bytes read

Алгоритм чтения файла "буферами":

-пытаться читать BufCount, прочесть Count
- обработать Count
- если Count<>BufCount, то закончить, иначе продолжить

```
bufsize equ 4096

buf db bufsize dup(?)
handle1 dw ?
.....................................
mov ah, 3Fh
mov bx, handle1
mov cx, bufsize
mov dx, offset buf
int 21h
jc ErrorRead
; ax = сколько прочлось
.............................
```

Писать в файл

INT 21 - DOS 2+ - WRITE TO FILE WITH HANDLE

AH = 40h
BX = file handle
CX = number of bytes to write
DS:DX = pointer to buffer

Return: CF = 1 if error
AX = Error Code
CF = 0 successful
AX = number of bytes written

Note: if CX is zero, no data is written, and the file is truncated or extended to the current position

"Если указать CX=0, то запись данных не производится, а размер файла устанавливается равным текущей позиции (последующее содержимое файла отсекается, если оно есть)."

В ряде ЯВУ такая функция зовется Truncate.

Переместиться к позиции

INT 21 - DOS 2+ - MOVE FILE READ/WRITE POINTER (LSEEK)

AH = 42h
AL = method value
0 = offset from beginning of file
1 = offset from present location
2 = offset from end of file
BX = file handle
CX:DX = offset in bytes

Return: CF = 1 if error
AX = Error Code
CF = 0 successful
DX:AX = new offset

```
;Перейти в конец файла
pos dd ?
.....................................
mov ah, 42h
mov al,2 ; с конца!
xor cx,cx
xor dx,dx ; dx:cx = 0
int 21h
jc ErrorPos
mov word ptr pos, ax
mov word ptr pos+2, dx
```

Спасибо за внимание.