**Лекция №4. Системные таблицы И ОБЛАСТИ ДАННЫХ, МЕТОДЫ РАБОТЫ С ФАЙЛАМИ.**

***Методы работы с файлами. Метод дескрипторов, DM.***

Метод дескрипторов предполагает использование дескрипторов (файловых индексов, описателей, манипуляторов), которые можем рассматривать как номера открытых файлов.

Процедура обращения к файлу в общем случае распадается на следующие операции:

1. Создание файла с заданным именем в указанном каталоге или открытие файла, если был создан ранее.
2. Запись в файл или чтение из файла всего сохраненного или любой его части.
3. Закрытие файла.

Открывая файл, назначает ему свободный элемент (блок описания файла). Этот элемент содержится в специальной системной таблицы - таблицы открытых файлов - SFT (SYSTEM FILE TABLE), которая располагается в оперативной памяти среди системных областей данных. Объем этой таблице определяется на этапе конфигурирования директивой Files в Config .sys.

Найдя в системе каталогов диска запись о открываемого файла, система записывает в выделенной ему SFT основные характеристики файла: имя, длина, атрибуты, дату и время создания, стартовый кластер и другие. Часть информации переписывается в элемент SFT по записи каталога, часть (указатель на ВРВ) ОС поставляет сама.

* SFT является двосливна ячейка, в которой хранится указатель файла - № байта относительно начала файла, с которого начинается очередная операция чтения или записи. Это позволяет организовать прямой доступ к файлу. Посылка на № блока описания файла в SFT возвращается в программу в виде дескриптору. Обращение к открытого файла выполняется по присвоенному ему дескриптору. Неоткрытый файл дескрипторов нет. Содержание в SFT модифицируется при работе с файлом, так сказать всегда отображает текущее состояние файла. После окончания работы с файлом его нужно закрыть. В процессе закрытия выполняется сброс буферов на

диск, модификация записи каталога и освобождения блока описания файла в SFT вместе с закрепленным за ним дескриптором. Таким образом, система может работать с неограниченным числом файлов, но количество одновременно открытых файлов определяется объемом системной таблицы файлов.

Для работы со стандартными устройствами система предоставляет 5 определенных дескрипторов:

* 0 стандартный ввод (CON)
* 1 стандартный вывод (CON)
* 2 стандартная ошибка (CON)
* 3 стандартный вспомогательный порт (AUX)
* 4 принтера (PRN).

Разница между дескриптором 1 и 2 состоит в том, что стандартный вывод (1)

можно перенаправить, а (2) всегда выводится на дисплей.

Система использует в своей работе ряд системных таблиц, которые в

документации отражены лишь частично.

Основной таблицей есть список списков (List of lists).

Адрес списка списков можно получить с помощью функции 52h прерывания

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 21h. | | функция | |  | возвращает | двосливну | адрес | список | списков | |
| в регистрах ES: BX. | | | |  |  |  |  |  |  |  |
|  |  |  |  | Табл. 1.13 Содержание некоторых полей списка списков | | | |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |
|  | смещение |  | Размер |  |  | описание | |  |  |  |
|  | -06h |  | 4 |  | Указатель на текущий диск буфера | | |  |  |  |
|  | -02h |  | 2 |  | Сегментный адрес первого блока МСВ | | |  |  |  |
|  | + 00h |  | 4 |  | Указатель на первый блок параметра диске | | |  |  |  |
|  | + 04h |  | 4 |  | Указатель на первую системную таблицу файлов (SFT) | | | |  |  |
|  | + 08h |  | 4 |  | Указатель на заголовок активного устройства CLOCK $ | | | |  |  |
|  | + 0Сh |  | 4 |  | Указатель на заголовок активного устройства CON | | | |  |  |
|  | + 10h |  | 2 |  | Максимальный размер сектора в байтах в блочном устройстве | | | | |  |
|  | + 12h |  | 4 |  | Указатель на информационный запись дискового буфера | | | |  |  |
|  | + 16h |  | 4 |  | Указатель на массив структур текущих каталогов | | | |  |  |
|  | + 20h |  | 1 |  | Число установленных блочных устройств | | |  |  |  |
|  | + 21h |  | 1 |  | Число элементов в массиве структур текущих каталогов | | | |  |  |
|  | + 22h |  | 18 |  | Заголовок драйвера устройства | |  |  |  |  |
|  | + 3Fh |  | 2 |  | Значения х в BUFFERS = x, y | |  |  |  |  |
|  | + 41h |  | 2 |  | Значение в в BUFFERS = x, y | |  |  |  |  |
|  | + 43h |  | 1 |  | Дисковод загрузки (1 = А :) | |  |  |  |  |
|  | + 44h |  | 2 |  | Размер расширенной памяти в килобайтах | | |  |  |  |

* байтах 4-7 списке списков хранится адрес SFT. По умолчанию при загрузке OC формируется одна SFT, в которую входит 5 блоков описания файлов.

При включении в файл Config.sys директивы FILES создается новая таблица, связанная с первой. Общее количество блоков в двух таблицах равна параметру директивы FILES.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Табл. 1.14 Блок описания файла системы |  |
|  |  |  |  |
| смещение | Размер | описание |  |
| 00h | 2 | Количество дескрипторов, закрепленных за данным файлом или 0, |  |
| если блок свободен |  |
|  |  |  |
|  |  |  |  |
| 02h | 2 | Режим доступа к файлу, заданный при его открытии |  |
|  |  |  |  |
| 04h | 1 | атрибуты файла |  |
|  |  |  |  |
| 05h | 2 | Информационное слово устройства |  |
| 07h | 4 | Указатель на заголовок драйвера символьного устройства или |  |
| указатель на блок параметров диска |  |
|  |  |  |
|  |  |  |  |
| 0Вh | 2 | Номер первого кластера файла |  |
|  |  |  |  |
| 0Fh | 2 | Дата последней модификации файла |  |
|  |  |  |  |
| 0Еh | 4 | Размер файла в байтах |  |
| 15h | 4 | Текущее положение указателя файла |  |
|  |  |  |  |
| 19h | 2 | Относительный номер последнего кластера файла прочитанного или |  |
| записанного |  |
|  |  |  |
|  |  |  |  |
| 1Вh | 4 | Номер сектора с записью каталога о данный файл |  |
|  |  |  |  |
| 1Fh | 1 | Номер записи каталога внутри сектора |  |
| 20h | 11 | Имя и расширение файла (без пути) |  |
|  |  |  |  |
| 2ЕH | 2 | Сегментный адрес PSP программы, которая открыла файл |  |
|  |  |  |  |
| 35h | 2 | Абсолютный номер последнего записанного или прочитанного |  |
| кластера |  |
|  |  |  |
|  |  |  |  |

Блок параметров диска (Drive Parameter Block, DPB) содержит информацию, частично повторяя данные в секторе загрузки. Адрес этого блока размещается в блоке описания файла.

Каждая SFT начинается с 6-ти байтового заголовка:

1. Оползень - 0, Размер - 4, Описание - адрес следующей таблицы или FFFFh в первом слове, если эта таблица последняя.
2. Смещение - 4, Размер - 2, Описание - Количество блоков описания файлов в данной таблице. Далее следуют блоки описания файлов.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Табл. 1.15 Блок параметров диска DPB |  |
|  |  |  |  |
| смещение | Размер | описание |  |
|  |  |  |  |
| 00h | 1 | Номер диска 00Н-А, 01Н-В ... |  |
|  |  |  |  |
| 01h | 1 | № устройства внутри драйвера устройства |  |
|  |  |  |  |
| 02h | 2 | Размер сектора в байтах |  |
| 04h | 1 | Наибольший номер сектора в кластере (размер кластера -1) |  |
|  |  |  |  |
| 05h | 1 | Число оползней, чтобы превратить кластеры на сектора |  |
| 06h | 2 | Число зарезервированных секторов с начала диска |  |
|  |  |  |  |
| 08h | 1 | число FAT |  |
|  |  |  |  |
| 09h | 2 | Число элементов в корневом каталоге |  |
| 0Вh | 2 | Номер сектора, с которого начинаются данные |  |
|  |  |  |  |
| 0Dh | 2 | Наибольший номер кластера (число кластера данных + 1) |  |
| 0Fh | 2 | Число секторов в одной FAT |  |
|  |  |  |  |
| 11h | 2 | Номер первого сектора каталогов |  |
|  |  |  |  |
| 13h | 4 | Адрес заголовке драйвера устройства |  |
| 18h | 1 | 00, если к диску было обращение, FF в противном случае |  |
|  |  |  |  |
| 19h | 4 | Указатель на следующий DРВ |  |
| 1Dh | 2 | Номер кластера, с которого нужно начинать поиск свободного |  |
| пространства на диске |  |
|  |  |  |
|  |  |  |  |
| 1Fh | 2 | Число свободных кластеров на диске, FFFFh-если неизвестно |  |
|  |  |  |  |

Для получения адреса блока параметров диска можем пользоваться функцией 32h. Она имеет следующий формат вызова:

Вход: AH-32h

DL-номер дисковода, 0- текущий.

выход: AL-0, если был задан верно номер дисковода, 0FFh- неверно.

DS: BX-адрес блока параметров диска.

Для получения адреса блока параметров текущего диска можем пользоваться функцией 1Fh.

Потому что порядок описания открытых файлов в SFT неизвестный заранее, для нахождения конкретного блока описания файла нужно:

1. Открыть файл с функцией 3Dh.
2. Получить от системы его дескриптор.
3. По дескриптору найти подходящий ему блок описания файла в SFT.

* приставке программного сегмента (PSP) является таблица файлов заданий (Job File Table, JFT) рисунок 1.10. По умолчанию эта таблица начинается с байта 18h PSP и имеет размер 20 байтов. В биты JFT по мере открытия файлов записывает номер соответствующих блоков описания файлов в SFT. Свободные байты JFT имеют коды FFh.

Относительные номера байтов JFT от ии начала и является дескрипторы открытых файлов. Первые 5 дескрипторов (от 0 до 4) система закрепляет за стандартными устройствами CON, AUX, PRN. Этим устройствам соответствуют 3 блока описания файлов в первой SFT.

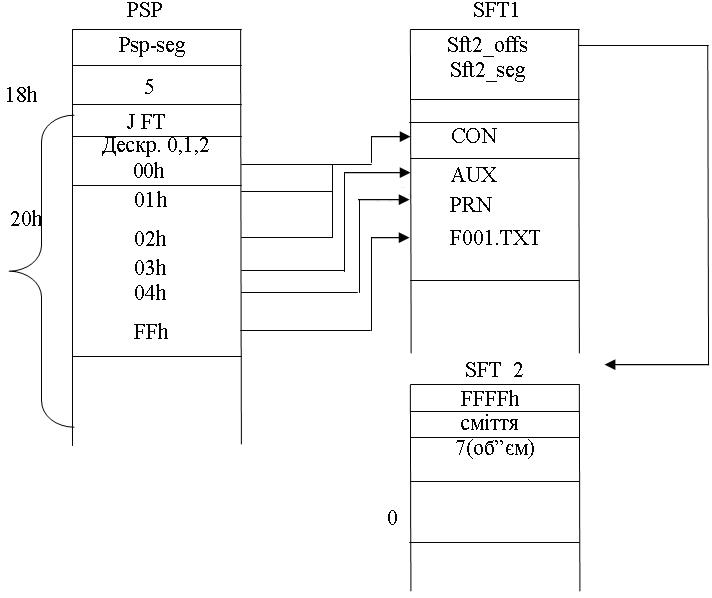


Рис. 1.10 Порядок описания открытых файлов

Таким образом первый свободный дескриптор должен иметь значение 5, а соответствующая ему информация о файле находится в блоке описания файла с номером 3. Пусть в CONFIG.SYS FILES = 12, открытые файлы F.001 и F.002.

* PSP программы кроме таблицы файлов задач есть еще поля, связанные с этой таблицей. В слове, имеет смещение 32h от начала PSP сохраняется размер JFT (по умолчанию 20 = 14h). В двосливний ячейке со смещением 36h - сегментный адрес PSP, которая при загрузке программы поступает в регистры DS и ES.

Поскольку размер JFT по умолчанию составляет 20 байтов причем 5 из них заняты конкретными значениями, то программа не может открыть более 15 файлов, чтобы открыть большее количество файлов нужно с помощью 67h создать новую JFT. Система, выделив для нее память, запишет в соответствующие PSP новый адрес и размер этой таблицы, скопируйте содержание старой JFT в новую таблицу и теперь будет открывать файлы с помощью новой JFT.

***Системные функции для работы с файлами методом дескрипторов.***

Все функции удобно разложить на смысловые группы.

1. Создание, открытие и закрытие файлов: - 3Ch создать файл;

- 5Ah создать временный файл; - 5Bh создать новый файл;

- 3Dh открыть файл; - 3Eh закрыть файл;

- 68h сбросить файл на диск.

1. Запись и чтение данных:

* 42h установить указатель;
* 40h записать в файл или в устройство
* 3Fh читать из файла или с устройства.

3. Изменение характеристик файла:

* 43h получить или установить атрибуты файла;
* 56h переименовать файл;
* 57h получить или переустановить дату и время создания файла.

4. Поиск файла:

* 1Ah установить адрес области передачи данных (DTA)
* 2Fh получить адрес области передачи данных
* 4Eh найти первый файл;
* 4Fh найти следующий файл.

1. Операция над каталогами: - 39h создать каталог;

- 3Аh выделить каталог;

- 3Bh изменить текущий каталог; - 47h получить текущий каталог.

1. Операции над дисками:

* 19h получить текущий диск;
* 0Eh изменить текущий диск;
* 36h получить информацию о диске.

Функция 3Ch и 5Bh позволяют создать файл с заданной спецификацией. Спецификация файла указывается в виде символьного строке, который завершается двоичным нулем. Адрес этой строки заносится в регистры DS: DX.

* регистре СХ задается код атрибутов файла создается.
* регистр АХ возвращается дескриптор созданного файла, который потом можем использовать для чтения файла или записи в файл.

Расхождение между функциями 3Ch и 5Bh проявляется в том случае, когда файл который создается уже существует.

Функция 3Ch уничтожает файл, который уже есть, а функция 5Bh завершается с CF = 1.

Функция 5Аh используется для создания временного файла, имя которому дает система. В регистрах DS: DX указывается адрес пути к файлу (но не имя файла) в виде символов строке, в конце которого должны быть предусмотрены 13 пустых байтов, куда поместит "/" и имя файла создаваемого которое заканчивается двоичным нулем . Можем предоставить любые атрибуты, кроме атрибутов отметки диске. При завершении программы автоматически такой файл не удаляется. Забота об этом лежит на программисты. Для записи в созданный временный файл нужно использовать дескриптор, возвращаемых функцией 5Ah в регистре AX.

Функция 3Dh позволяет открыть уже существующие файлы. В регистрах DS: DX задается спецификация файла в виде строке ASCIIZ, в регистр AL - режим доступа (0-чтения, 1 запись, 2 чтение и запись). В дальнейшем запись в файл и чтение из него осуществляется с помощью дескриптору, возвращаемого функцией в регистр AX.

Для каждого открытого файла создается и поддерживается указатель, который представляет собой относительный номер байта в файле, начиная с которого будет осуществляться запись или чтение данных. Указатель только что открытого или созданного файла устанавливается системой на начало файла. Функции чтения или записи смещают его в число прочитанных или записанных байтов.

Для организации прямого доступа к произвольному места файла предусмотрена функция 42h. Она позволяет задать положение указателя относительно начала файла (AL = 0), конца файла (AL = 2) или текущего положения указателя (AL = 1).

Само значение положения указателя (со знаком) заносится в регистр CX (старшая половина) и DX (младшая).

Функция 3Fh используется для чтения из файла или устройства, 40h-записи. Перед вызовом функции в регистр BX помещается дескриптор, в регистр CX число байтов читаемых или записываются, а в регистры DS: DX - адрес буфера в программе пользователя.

Иногда возникает необходимость найти в каталоге все файлы, которые удовлетворяют условиям шаблона (например, \*. Txt). Такой поиск осуществляется функциями 4Eh и 4Fh (найти следующий файл). Для их использования необходимо с помощью функции 1Аh организовать в программе область передачи данных (DTA, Disk Transfer Area), размером не менее 64б или с помощью функции 2Fh получить адрес DTA. В качестве такой DTA Система использует PSP от байта 80h до конца. Система помещает в DTA информацию о найденном файл (атрибут, размер, время, дата создания и т. Д.) В байтах 1Eh ... 2Fh в DTA содержится имя и расширение файла в виде строки ASCIIZ.

При поиске файлов при заданном шаблоне сначала активизируется функция 4Eh.

* регистрах DS: DX содержится адрес строки ASCIIZ с дорогой рассматриваемого каталога. В регистре CX- код комбинации возвращает в CF = 0, а имя и расширение файла в виде строки ASCIIZ помещается в DTA. Получив имя файла, можно открыть его с помощью функции 3Dh и выполнить операции над ним.

Функция 4Fh используется так же, как и 4EH.

***Метод FCB для работы с файлами.***

FCB - блок управления файлом. Используется традиционными функциями.

Расширенные функции используют метод ФМ - файловый манипулятор.

Основное в этом методе заключается в том, что пользовательская программа должна построить блок управления файлом (FCB) для каждого файла, с которым она работает. FCB представляет собой область памяти которая используется с длиной 37 или 44 байта, которая имеет стандартный формат. В FCB поддерживается необходимая для работы с файлом информация: устройство, имя, размер, дата, размер записи, текущая позиция в файле и т. Д.

Отметим, что не предусмотрено место пути к файлу - доступные только файлы текущего каталога.

Часть информации (устройство, имя) заполняется программой еще до открытия файла. При открытии файла заполняет другие поля FCB (размер, дата и т. Д.). Пользовательская программа задает параметры для системных функций в полях FCB, а они со своей стороны возвращают информацию через FCB. В этом процессе часто допускают ошибки, а создание и поддержка FCB для каждого файла требует дополнительных усилий. Одна из типичных ошибок - использование одного FCB для работы

* несколькими файлами, без выполнения CLOSE при каждой смене имени файла. Ошибкой считается также устранения или переименование открытого файла.

При этом методе обмен данными между программой и файлами всегда через буфер или так называемую область обмена с диском (DTA - Disk Transfer Area). Адрес буфера поддерживается в указателе к DTA. При запуске программы, указатель к DTA указывает на область длиной 128Б в PSP. В дальнейшем программа может изменить (с помощью специальных системных функций) указатель к DTA.

Необходимость в этом может возникнуть, если например 128Б недостаточно. Конечно, по новому адресу нужно иметь свободное место соответствующего размера для нужной DTA.

Описание FCB.

При рассмотрении системных функций нужно знать поля FCB. FCB делится на две основные части: главная часть размером 37 байтов и префикс размером 7 байтов. В главной части содержится основная информация о файле - устройство, имя, размер записи, ее номер и др.

Префикс используется при необходимости отметить специальные атрибуты файла (скрытый, системный и т.д.). Все системные функции при методе FCB получают в качестве параметра адрес данного FCB. Если со смещением 0 от этого адреса находится байт с содержанием FFh, это означает, что FCB имеет префикс, и смещение для таких полей FCB растет соответственно на 7 б. Если байт со смещением 0 имеет значение, отличное от FFh, система считает, что FCB не имеет префикса и принимает это значение как номер устройства. Номер устройства никогда не может быть FFh.

Поля FCB.

1. Смещение (0). Индикатор активного префикса FCB. Значение FFh означает активный префикс - наличие специальных атрибутов. Записывается к выполнению OPEN.
2. Смещение (1) .Резервовано. Должно содержать 00h.
3. Смещение (6). Задает атрибуты файла: скрытый, системный, отметка диске или каталог. Для обычных файлов и файлов только для чтения не нужен. Заполняется к выполнению OPEN.
4. Смещение (7). Номер устройства. Перед открытием файла пользователь записывает в данное поле номер устройства, с которым хочет работать. Используется нумерация, при которой 0 означает текущее устройство, 1 устройство А и т. Д. Если использовано значение 0, система открывает файл на текущем устройства и записывает в поле его действительный номер. Заполняется к выполнению OPEN.
5. Смещение (8) и (16). Имя и расширение файла. Поля выровнены влево и дополнены справа промежутками. Можно использовать малые и большие буквы. Резервированные имена (CON, AUX, COM1, COM2, PRN, LPT1, LPT2, NUL) интерпретируются как устройства, а не как имена файлов. Заполняются до выполнения OPEN.
6. Смещение (19) и (39). Номер текущего блока и номер записи в блоке. Нумерация начинается с 0 Каждый блок состоит из 128 записей, чей размер определяется полем со сдвигом (21). Эти два поля используются при последовательном доступе к файлу. При открытии файла поля инициализируются. Пользователь инициализирует их перед первой операцией (последовательное чтение или запись 0, а система автоматически увеличивает их после каждой операции. В данном случае последовательный доступ реализован как вариант прямого. В любой момент пользователь может позиционироваться в произвольном месте файла и начать с этого места последовательные операции.
7. Смещение (21). Размер логической записи файла. Используется как при последовательном так и при прямом доступе. В самом файле, записанном на диску, не

сохраняется никакой информации о размере логической записи файла просто последовательностью байтов. Размер логической записи в FCB отражает, как рассматривается логическая структура файла в конкретной программе. Очевидно, тот же файл можно обрабатывать, задав разный размер записи. Во многих случаях, когда файл не состоит из записей фиксированной длины (например, текстовый файл), удобно работать с длиной записи 1 байт. При открытии файла система заполняет

длину записи 128Б. Если нужна другая длина, ее следует записать в поля 23, 27, 29, 40 FCB после открытия файла.

***Выполнение программ. Структура и загрузки EXE i COM файлов.***

EXE файлы.

EXE файлы создаются редактором связей LINK и состоят из двух частей: приставка и модуль исполнения. Приставка состоит из управляющей информации и таблицы символов перемещаемых. Модуль выполнения содержит инструкции и данные программы и находится непосредственно за префиксом. В модуле выполнения можно иметь адресные константы, которые нужно настроить для конкретного адреса загрузки. Это адреса сегментов, значение которых в модуле вычисленные относительно его начала. Во время выполнения адресные константы должны содержать абсолютные адреса. Преобразование относительных адресов на абсолютные выполняются системной программой загрузки, использует информацию из таблицы символов перемещаемых - там описано местоположение адресных констант в модуле исполнения.

Управляющая информация в приставке.

Поле 1. Смещение 00h. 4Dh, 5Ah - идентификатор EXE файла.

Поле 2. Смещение 02h. Длина файла по модулю 512 (остаток деления длины файла на 512)

Поле 3. Смещение 04h. Длина файла в блоках по 512б.

Поле 4. Смещение 06h. Количество элементов в таблице символов перемещаемых.

Поле 5. Смещение 08h. Размер приставки в параграфах.

Поле 6. Сдвиг 0Аh. Минимальное количество параграфов, необходимых после загруженной программы (обычно 0).

Поле 7. Смещение 0Сh. Максимальное количество параграфов, необходимых после загрузки программы (FFFFh для программ, загружаемых в младшие адреса памяти, 0000h- в старшие адреса памяти)

Поле 8. Смещение 0Еh. Смещение SS в модуле исполнения (в пунктах).

Поле 9. Смещение 10h. Значение SP при получении управления в модуле исполнения. Поле 10. Смещение 12h. сумма (отрицательная сумма всех слов в файле с игнорированием

перегрузки).

Поле 11. Смещение 14h. Значение IP, когда модуль выполнения получает управление.

Поле 12. Смещение 16h. Смещение СS в модуле исполнения (в пунктах).

Поле 13. Смещение 18h. Начало таблицы символов перемещаемых (сдвиг с начала файла в байтах)

Поле 14. Смещение 1Ah. Номер перекрытия (0-для резидентной части программы) Адрес таблицы символов, перемещаемых находится в поле 13, а количество

* элементов - в поле 4. Каждый элемент таблицы дает смещение адресной константы в модуле исполнения, который нужно настроить относительно адреса загрузки, перед тем, как передать управление модулю. Элементы имеют длину 2 слова и формат сегмент: смещение (сдвиг в первом слове).

Загрузка программы осуществляется в следующей последовательности:

* 1. Управляющая информация с префикса считывается в рабочую область системы.
  2. Определяется размер модуля выполнения в зависимости от значений полей 2,3,4,5.
  3. Находится первый свободный блок памяти, размер которого достаточен для модуля исполнения и PSP. Блок присваивается программе. Если программа загружается в старшей области памяти (поле 7), то ищется последний, а не первый свободный блок памяти необходимого размера. Программа загружается в конце, а не в начале блока, часть блока может остаться свободной.
  4. Строится PSP для программы. Сегмент после PSP называется старшим сегментом. Модуль выполнения загружается с начала этого сегмента.
  5. Из таблицы символов перемещаемых считывается в рабочую область. К каждому элементу добавляется значение старшего элемента. Полученная адрес в формате сегмент: смещение указывает слово модуля выполнения (в памяти), к содержанию которого добавляется значение старшего сегмента. После окончания этого процесса адресные константы в модуле выполнения загружены в соответствии с адресом загрузки (стартового сегмента).
  6. Регистрам ES i DS присваивается значение сегмента программы (сегментный адрес PSP). SS и SP присваиваются значения полей 8 и 9 приставки и добавляется стартовый сегмент. Значение CS образуется как сумма поля 12 в приставке стартового сегмента. IP образуется из поля 11.

1. Теперь уже модуль выполнения настроен (перемещен) в соответствии с адреса загрузки. Управление передается по адресу CS: IP.

СОМ файлы.

СОМ файлы не имеют префиксов. Они построены так, чтобы не содержали адресных констант, зависимых от адреса загрузки программы в памяти (все в программе подано, как смещение с начала сегмента для кода, включая данные и стек. Следовательно, размер программы не может превышать 64К). СОМ представляет собой точную копию программы в двоичном виде, в котором ее нужно загрузить в память.

Поэтому загрузки СОМ-файлов сводится к определению большого свободного блока памяти, построении PSP и считывания всего файла в область после PSP.

Регистрам CS, DS, SS, ES присваивается значение сегмента программы (сегментный адрес PSP). Указатель инструкции IP получает значение 100h. SP получает значение FFFEh, то есть для стека используется конец адресного пространства с 64К. Байт по адресу FFFEh в стеке содержит 00h, так что после конца команды RET, в конце программы, в IP получается значение 00h, то есть выполняется инструкция INT 20h в начале PSP.

Преобразование ЕХЕ файлов на СОМ файлы.

Преобразование осуществляется вспомогательной командой EXE2BIN.ЕXE. Командная строка имеет формат EXE2BIN <входной файл> [<выходной файл>].

Входной файл должен быть в формате ЕХЕ. Он имеет разрешение по умолчанию ЕХЕ. Общая длина инструкций и данных не может превышать 64 Кб. Не допускается сегмент типа STACK. Исходные файлы имеют формат СОМ (копия программы в двоичном виде) и по умолчанию получают расширение ВИN.

Возможны два способа преобразования в зависимости от значения CS: IP в приставке ЕХЕ файла (поля 12 и 11):

1. Значение CS: IP-0: 0. Это означает, что сегмент кода является первым в программе и первая инструкция - ORG 0 (или вообще нет инструкции ORG). Если в программе есть адресные константы, которые нужно настроить в соответствии с конкретного адреса загрузки, это осуществляется при преобразовании. EXE2BIN поставит вопрос об абсолютной адрес загрузки программы и выполнит настройки в соответствии с полученной ответа. Итак, для правильного выполнения программа должна быть

загружена именно с этого адреса. Невозможно использовать систему для загрузки таких программ - их должны загружать (считывать) конкретные пользовательские программы. В этом случае PSP не строится.

1. Значение CS: IP дает смещение 100h с начала выполнения модуля. Это означает, что программа будет выполнять как нормальный СОМ файл с построением PSP и инструкциями непосредственно после него. В этом случае программа не должна содержать адресных констант, которые нужно настроить, то есть таблиц символов которые

перемещаются в приставке ЕХЕ файла должна быть пустой (поле 4 = 0). Полученный СОМ файл можно скачать в любое место памяти.

Если ЕХЕ файл не соответствует перечисленным ограничениям, EXE2BIN выдает сообщение "File cаnnot be convertеd" (файл невозможно превратить) и заканчивает работу. Второй способ предпочтительнее, так как создаются главные СОМ файлы.

Правила создания таких файлов:

1. Программа должна начинаться директивой ORG 100h и входная точка должна быть в начале программы.

ORG 100h START:

.................. ..

END START

После ее загрузки CS содержит сегментный адрес PSP, а IP установлен в значеня 100h. Если нужно начать выполнение с другого места, первой инструкцией после ORG 100h должна быть JMP.

1. Не допускается определить отдельные сегменты для данных и для стека. Определяется только сегмент для кода. Используется директива ASSUME, чтобы оставить все сегментные регистры сегмента для кода.
2. Данные можно расположить в любом месте программы, чтобы не путать их с

кодом.

1. НЕ допускают инструкции, имеющие в качестве операндов сегментные адреса. В частности, в начале программы отсутствуют определения отдельного сегмента данных, является типом для ЕХЕ файлов.
2. При загрузке программы SS, как и другие сегментные регистры, имеет значение сегментного адреса PSP. SP указывает последнюю адрес 64К байтового адресного

пространства (например, программа, остается резидентной). Можно изменить SP еще до использования стека.

СОДЕРЖАНИЕ PSP.

Поле 1.Длина 2, смещение 0 Содержит код инструкции INT 20h. Прерывания 20h - одно из средств завершения выполнения программы. Итак, программа может завершаться инструкцией для перехода на начало PSP. СS должен содержать сегментный адрес PSP. Эта возможность выглядит вполне бессмысленной при наличии системной функции для завершения программы - 4Ch. Не говоря о том, что программа сама может выполнить прерывание 20h. На самом деле INT 20h в начале PSP имеет другое назначение - повышает надежность при выполнении программы с нерешенными внешними связями. Если программа выполняет переход к адресу, не определена при редактировании, то это приведет к ее завершению. Редактор связей дает значение 0 всем таким адресам, то есть будет осуществлен переход к началу PSP.

Поле 2.Длина 2, смещение 2. Содержит размер имеющейся оперативной памяти в блоках по 16 байт (пунктах). Точнее, поле содержит сегментный адрес последнего параграфа памяти. Это значение использует программа CHKDSK в своем отчете. Нужно знать, что в некоторых случаях это значение может не совпадать с действительной количеством памяти. Например, некоторые виртуальные диски используют старшие адреса памяти и меняют информацию о последней адрес. Поэтому может существовать разница между полем 2 PSP и значением, полученным прерыванием 12h BIOS, что вытягивает информацию об имеющейся память.

Поле 3. Длина 1, смещение 4. резервируемой.

поле 4. Длина 5, смещение 5. Дальний переход (FAR JMP 000C0h) дает возможность выполнить некоторые уже существующие программы, использующие другой метод обращения к системным функциям. При этом методе номер функции записывается в CL вместо AL, другие параметры задаются в соответствии с описанием функции. Затем выполняется JMP (в текущем сегменте программы). Затем выполняется требуется, после чего управление возвращается программе. АХ всегда теряется. Поскольку новые программы, применяют этот метод, почти не применяются, практически поле 4 не интересно как средство для выполнения системных функций. С другой стороны, второе слово поля содержит полезную информацию, которую можно использовать для других целей, например, как

индикатор, обладающий сегментом длины не менее 64К. Если он меньше, то поле содержит его размер.

Поля 5,6,7. Длины 4, оползни А, Е, 12 содержат векторы прерываний, которыми они

были к запуску программы. Это INT 22h (адрес завершения), INT 23h (прерывание программы с помощью <Ctrl / C>), INT 24h (обработка критических ошибок). Программа может изменять эти векторы как угодно. При ее завершении система восстанавливает их предыдущий содержание с РSP. Например, программа может создать собственный обработчик <Ctrl / C>, но после ее завершения будет установлена ​​стандартная обработка. Если программа изменяет некоторые из этих полей в РSP, изменения останутся и после ее завершения.

Поле 8. Длина 2, смещение 16 сегментный адрес родительского РSP, таблица файлов

задача JFТ.

Поле 9.Длина 2, смещение 2С содержит адрес окружения программы. указанная

только адрес сегмента, смещение = 0.

Поле 10.Длина 24 смещение 2Е. рабочая область

2E, 4б - SS: SP программы при входе в последний вызов INT 21h,

32h, 2б - число элементов в JFT (по умолчанию 20),

34h, 4б - адрес JFT (по умолчанию PSP: 018h),

40h, 2б - версия в формате функции 30h.

Поле 11.Длина 3, смещение 50 содержит инструкции INT 21h i RET FAR, что дает возможность выполнять косвенно обслуживающие функции системы. Для выполнения этой возможности программа должна подготовить все необходимое для использования нужной системной функции и затем осуществлять дальний переход к PSP + 50h.

Поле 12. Длина 2, смещение 53. резервируемой.

поля 13,14,15,16**.** Длина 7,9,7,9 смещение 55,5С, 65,6С соответственно содержит префиксы

першогоидругогоFCB (поля13,15) исамипершийи

второй FCB (поля 14,16).

Эти поля помогают программам, использующих для работы с файлами метод FCB. Основная их цель - облегчить обработку параметров из командной строки программы. Предполагается, что эти параметры - имена файлов, обрабатываемых программой. Система конструирует с их помощью два неоткрытых FCB в PSP. Если они нужны программе, она может их открыть и использовать.

Возможные осложнения: два FCB перекрываются как друг с другом, так и с DTA в PSP, если используется только первый из них и изменится адрес DTA, то все хорошо. В противном случае, один или оба FCB нужно переместить с PSP до их открытия.

Поля 17,18.Длины 1 и 127, смещение 80,81. Поле 17 содержит длину параметров командной строки, поле 18 - параметры командной строки. Поля обеспечивают доступа к всех параметров командной строки.

Эти два поля перекрываются полем 19 - DTA, поэтому их нужно обработать до возможного разрушения.

Поле 19.Длина 128, смещение 80 используется как DTA по умолчанию. После извлечения необходимой информации программа может использовать часть PSP (от смещения 5С и до конца) как рабочую область.