Лабораторная работа № 13
Изучение возможностей различных файловых систем

Цель работы: изучить основные возможности различных файловых систем.
Оборудование: ПК, Интернет.
Время выполнения: 90 минут.

КРАТКАЯ ТЕОРИЯ И МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ:

Файловая система операционной системы MS-DOS

Файловая система *MS-DOS* представляет собой увеличенную и улучшенную версию файловой системы *CP/M*, которая работает только на платформах с центральным процессором *Intel*, не поддерживает многозадачности и работает только в реальном режиме *IBM PC*. Файловая система *MS-DOS* во многом напоминает файловую систему *CP/M,* включая использование имен файлов, состоящих из 8 + 3 символов верхнего регистра. В первой версии системы (*MS-DOS 1.0*) был всего один каталог, как и в *CP/M*. Однако, начиная с версии операционной системы *MS-DOS 2.0*, функциональность файловой системы значительно расширилась. Самым серьезным улучшением явился переход на иерархическую файловую систему, в которой каталоги могли вкладываться друг в друга на произвольную глубину. Это означало, что корневой каталог (размер которого по-прежнему был ограничен) мог содержать подкаталоги, которые так же могли содержать свои подкаталоги. Связи, принятые в *UNIX*, не допускались, поэтому файловая система представляла дерево, начинавшееся в корневом каталоге.

Прикладные программы часто начинают с того, что задают в корневом каталоге подкаталог, в который записывают свои файлы, что позволяет программам избежать конфликта. Так как сами каталоги хранятся в *MS-DOS* как файлы, нет ограничения на число каталогов или файлов на диске. Однако в отличие от *CP/M*, в *MS-DOS* нет концепции различных пользователей. Соответственно, любой вошедший в систему пользователь получает доступ ко всем файлам.

Чтобы прочитать файл, программа, работающая в системе *MS-DOS*, должна вначале сделать системный вызов *open*, чтобы получить *дескриптор* файла[1]. Системному вызову *open* в качестве одного из входных аргументов следует указать путь к файлу, который может быть как абсолютным, так и относительным (относительно текущего каталога). Файловая система открывает каталоги, перечисленные в пути, один за другим, пока не обнаруживает последний каталог, который считывается в оперативную память. Затем в считанном каталоге ищется описатель файла, который требуется открыть.

Хотя каталоги в файловой системе *MS-DOS* переменного размера, используемые каталоговые записи, как и в *CP/M*, имеют фиксированный размер 32 байт (рис.1.). Описатель файла содержит: имя файла, его атрибуты, дату и время создания, номер начального блока и точный размер файла. Имена файлов короче 8 + 3 символов выравниваются по левому краю полей и дополняются пробелами, каждое поле отдельно. Поле *Attributes* (атрибуты) представляет собой новое поле, содержащее биты, которые указывают тип файла (заархивирован, системный или скрытый) и действия, которые ему разрешены (чтение или чтение и запись). Запись в файл, для которого разрешено только чтение, не разрешается. Таким образом, осуществляется защита файлов от случайной записи или удаления.

Бит *archived* (архивный файл) не устанавливается и не проверяется операционной системой. Он зарезервирован в описателе для архивирующих программ уровня пользователя, сбрасывающих этот бит при создании резервной копии файла, в то время как программы, модифицирующие файл,

---

[1] Дескриптор файла (*file handle* – логический номер файла; индекс файла; описатель файла) – уникальный идентификатор, присваиваемый системой *Windows* файлу в момент его открытия или создания, и существующий до момента его закрытия.

устанавливают этот бит. Таким образом архивирующая программа определяет какие файлы подлежат архивации. Бит *hidden* (скрытый файл) позволяет не отображать файл в перечне файлов каталога, что позволяет скрыть от неопытных пользователей файлы, назначение которых им неизвестно. Бит *system* (системный) также скрывает файлы и защищает их от случайного удаления командой *del*, он установлен у основных компонентов системы *MS-DOS*.

Каталоговая запись также содержит дату и время создания или последнего изменения файла. Время хранится с точностью ±2 секунды, так как для него отведено 2-байтовое поле, способное содержать всего 65536 уникальных значений, а в сутках 86400 секунд. Поле времени разбивается на подполя: секунды (5 бит), минуты (6 бит) и часы (5 бит). Шестнадцатиразрядное поле даты также разбивается на три подполя: день (5 бит), месяц (4 бит) и год – 1980 (7 бит). При 7 двоичных разрядах для хранения года и 1980 в качестве точки отсчета, максимальное значение года, которое можно получить – 2107-й, поэтому файловая система *MS-DOS* имеет встроенную проблему 2108 года.

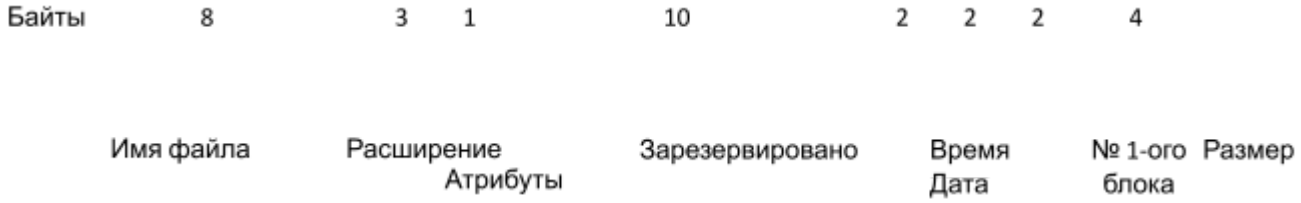
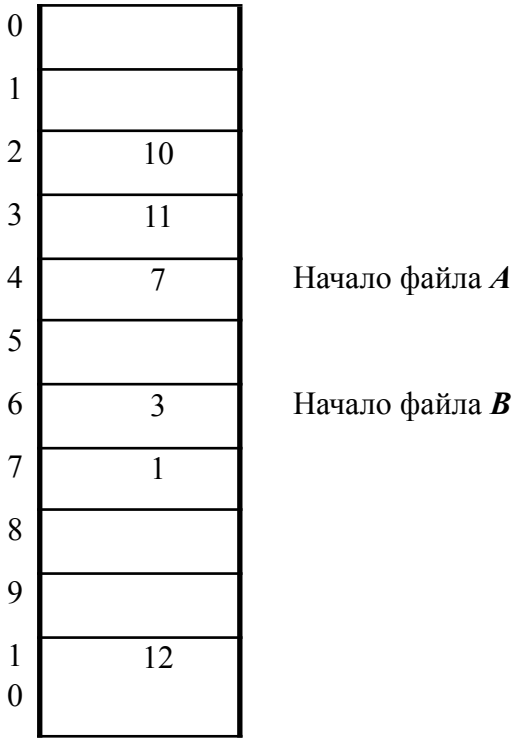


Рис. 1. Формат каталоговой записи в системе MS-DOS

В отличие от файловой системы *CP/M*, не хранящей точного размера файла, система *MS-DOS* хранит точный размер файла и номера блоков файла в специальной таблице размещения файлов (*FAT*), которая помещается в оперативную память (в *CP/M* дисковые адреса файлов хранятся в их описателях). В каталоговой записи файловой системы *MS-DOS* хранится номер первого блока файла, который используется в качестве индекса для 64 К[2] элементов *FAT*-таблицы. Все блоки файла могут быть найдены, если проследовать по цепочке элементов таблицы (рис.2.).



[2] Справедливо только для *FAT-16*

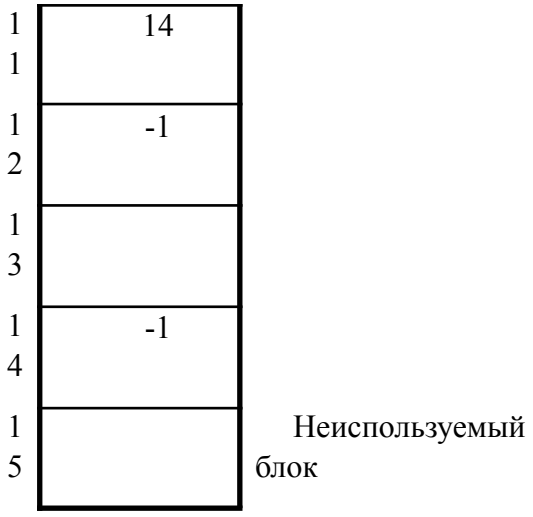


Рис. 2. Таблица размещения файлов

В зависимости от количества блоков на диске в системе *MS-DOS* применяется три версии файловой системы *FAT*: *FAT-12*, *FAT-16* и *FAT-32*. Во всех файловых системах *FAT* размер блока диска в байтах может быть установлен равным некоторому числу, кратному 512 с наборами разрешенных размеров блоков (*кластеров*), различными для каждого варианта *FAT*. В первой версии системы *MS-DOS* использовалась *FAT-12* с 512- байтовыми блоками, что позволяло создавать дисковые разделы размером до 212 х 512 байт. При этом максимальный размер дискового раздела мог составлять 2 Мбайт, а в оперативной памяти *FAT*-таблица занимала 4096 элементов по два байта каждый. Такая система хорошо работала на гибких дисках. Для работы на жестких дисках корпорация *Microsoft* решила использовать дисковые блоки (кластеры) размером 1,2 и 4 Кбайт, что позволило сохранить структуру и размер таблицы *FAT-12* и увеличить размер дискового раздела до 16 Мбайт.

Так как *MS-DOS* поддерживала до четырех дисковых разделов, файловая система *FAT-12* могла работать с дисками емкостью до 64 Мбайт. Для поддержки жестких дисков большего размера была разработана файловая система *FAT-16* с 16-разрядными дисковыми указателями. Дополнительно было разрешено использовать кластеры размеров 8, 16 и 32 Кбайт. Таблица *FAT-16* занимала 128 Кбайт оперативной памяти, максимальный размер дискового раздела, поддерживаемый файловой системой, составлял 2 Гбайт (64 К элементов по 32 Кбайт каждый), максимальный размер диска составлял 8 Гбайт (4 раздела по 2 Гбайт каждый).

Для второй версии операционной системы *Windows 95* была разработана файловая система *FAT-32* с 28-разрядными адресами. При этом версия системы *MS-DOS*, лежащая в основе *Windows 95*, была адаптирована для поддержки *FAT-32*. Размер разделов увеличился до 2 Тбайт (2048 Гбайт) и 8-гигобайтный диск мог состоять всего из одного раздела (при использовании *FAT-16* он должен был содержать четыре раздела, что представлялось пользователям как логические устройства: *C:, D:, E: и F:*). Кроме того, для дискового раздела заданного размера могли использоваться блоки меньшего размера, например, 4 Кбайт (для FAT-16 использовались 32-килобайтные блоки). При размере блока в 32 Кбайт даже маленький (10-байтовый файл) занимает на диске 32 Кбайт. Так как многие файлы имеют размер меньше 32 Кбайт, то при использовании 32-килобайтных блоков около 3/4 дискового пространства теряется, то есть эффективность использования диска низкая. При 8-килобайтных

файлах и 4-килобайтных блоках потерь дискового пространства нет, но значительно увеличился размер оперативной памяти, занимаемой таблицей FAT.

Файловая система операционной системы Windows 98

Первая версия операционной системы *Windows 95* использовала файловую систему *MS-DOS*, с именами файлов из 8 + 3 символов и системами *FAT-12* и *FAT-16*. Во второй версии *Windows 95* были разрешены длинные имена файлов в новой файловой системе *FAT-32*, разработанной для поддержки дисков размером более 8 Гбайт и дисковых разделов больше 2 Гбайт. В операционных системах *Windows 98* и *Windows Me* использовалась также файловая система *FAT-32*.

В *Windows 98* была разработана новая система поддержки длинных имен, обладавшая обратной совместимостью со старой системой имен 8 + 3, применявшейся в *MS-DOS*. Структура каталогов представляла собой список 32-байтовых описателей, формат которых был заимствован у файловой системы *CP/M* (написанной для процессора *Intel 8080*). Однако в 32-байтовом описателе файла оставались незадействованными 10 байт (рис.1), которые стали использоваться в *FAT-32* (рис.3.)

Рис. 3. Формат каталоговой записи в системе *Windows 98*

Изменение каталоговой записи состояло в добавлении пяти новых полей на место неиспользовавшихся 10 байт. Поле «*NT*» было предназначено для совместимости с *Windows NT* и обеспечивало отображение имени файла в правильном регистре. Поле «Время создания» (*Sec*) решило проблему невозможности хранения времени суток в 16-битовом поле с точностью до секунды. Восемь дополнительных разрядов позволили хранить поле «Время создания» (*Creation time*) с точностью до 10 мс. Поле «Дата последнего доступа» (Last access) было создано для хранения даты последнего доступа к файлу. В связи с переходом на файловую систему *FAT-32* для хранения старших разрядов номера начального блока файла потребовались дополнительные 16 бит.

Для предоставления длинным именам файлов обратной совместимости с *MS-DOS* было принято решение – назначение каждому файлу двух имен: длинного имени файла (в формате *Unicode*, для совместимости с *Windows NT*) и имени формата 8 + 3 для совместимости с *MS-DOS*. При создании файла, имя которого не удовлетворяло правилам *MS-DOS*, *Windows 98* создавало дополнительное имя формата *MS-DOS* в соответствии с определенным алгоритмом. Использовались первые шесть символов имени, которые при необходимости преобразовывались в верхний регистр *ASCII*, после чего к ним добавляется суффикс « ~ 1». Если такое имя уже было, то использовался суффикс « ~ 2) и т.д. Кроме того, удалялись пробелы и лишние точки, а определенные символы преобразовывались в символы подчеркивания. Например, имя файла «*The time has come the walms*» получало формат *MS-DOS* «*THETIM~1*».

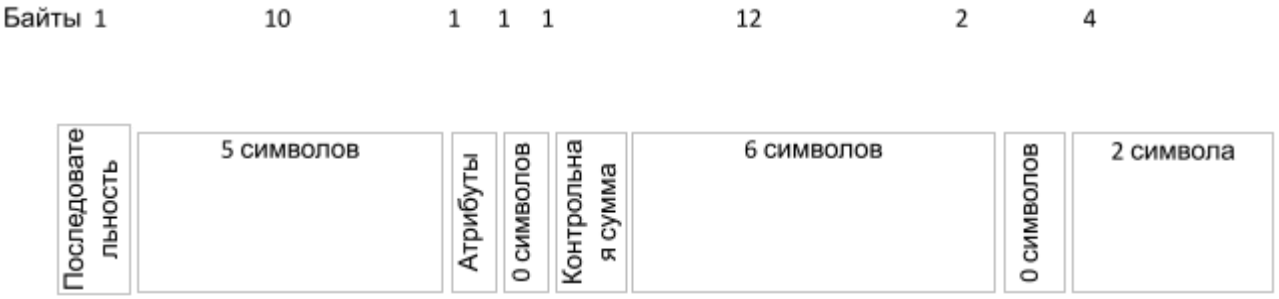


Рис. 4. Формат каталоговой записи с фрагментом
длинного имени файла в *Windows 98*

Имя формата *MS-DOS* хранилось в каталоге в описателе (рис. 3.). Если у файла было также длинное имя, оно хранилось в одной или нескольких каталоговых записях, предшествующих описателю файла с именем в формате *MS-DOS*. Каждая такая запись содержала до 13 символов формата *Unicode*. Элементы имени хранились в обратном порядке, начинаясь сразу перед описателем файла в формате *MS-DOS* и последующими фрагментами перед ним (рис.4.)

Для отличия каталоговых записей, содержащих длинные и короткие имена файла в поле «*Атрибуты*» (*Attributes*) для фрагмента длинного имени устанавливалось значение «*0x0F*», что соответствовало невозможной комбинации атрибутов для описателя файла в *MS-DOS*. Старые программы, написанные для работы в *MS-DOS*, при чтении каталога игнорировали такие описатели как неверные. Порядок фрагментов имени учитывался в первом байте каталоговой записи. Последний фрагмент имени отмечался добавлением к порядковому номеру числа 64. Поскольку для порядкового номера использовалось 6 бит, теоретически максимальная длина имени файла могла составить 63 x 13 = 819 символов. На практике она ограничивалась 260 символами.

Каждый фрагмент длинного имени содержал поле «*Контрольная сумма*» (*Checksum*) во избежание проблем с переводом длинных имен файлов в короткие имена. Реализация файловой системы *FAT-32* концептуально близка к реализации файловой системы *FAT-16*. Однако вместо массива из 65536 элементов в ней используется столько, сколько нужно, чтобы покрыть весь раздел диска. Если диск содержит миллион блоков, то и таблица будет состоять из миллиона элементов. Для экономии памяти система *Windows 98* не хранит их все сразу в памяти, а использует окно, накладываемое на таблицу.

Файловая система операционной системы UNIX

Операционная система *UNIX* представляет собой ядро многопользовательской операционной системы с разделением времени. Она дает пользователям возможность запускать свои программы, управляет периферийными устройствами и обеспечивает работу файловой системы.

Работу операционной системы *UNIX* можно представить в виде функционирования множества взаимосвязанных процессов. При загрузке системы сначала запускается ядро (процесс 0), которое в свою очередь запускает командный интерпретатор *shell* (процесс 1). Взаимодействие пользователя с системой *UNIX* происходит в интерактивном режиме посредством командного языка. Оболочка операционной системы *shell* интерпретирует вводимые команды, запускает соответствующие программы (процессы), формирует и выводит ответные сообщения.

Важной составной частью *UNIX* является файловая система, которая является сложной многопользовательской системой, так как в основе этой системы лежала операционная система *MULTICS*. Файловая система имеет иерархическую структуру, образующую дерево каталогов и файлов. Дерево начинается в корневом каталоге, с добавлением связей, формирующих направленный ациклический граф. Имена файлов могут содержать до 14 символов, включающих в себя любые символы *ASCII*, кроме косой черты (использовавшейся в качестве разделителя компонентов пути) и символа *NUL* (использовавшегося для дополнения имен короче 14 символов). Символ NUL обозначается байтом 0.

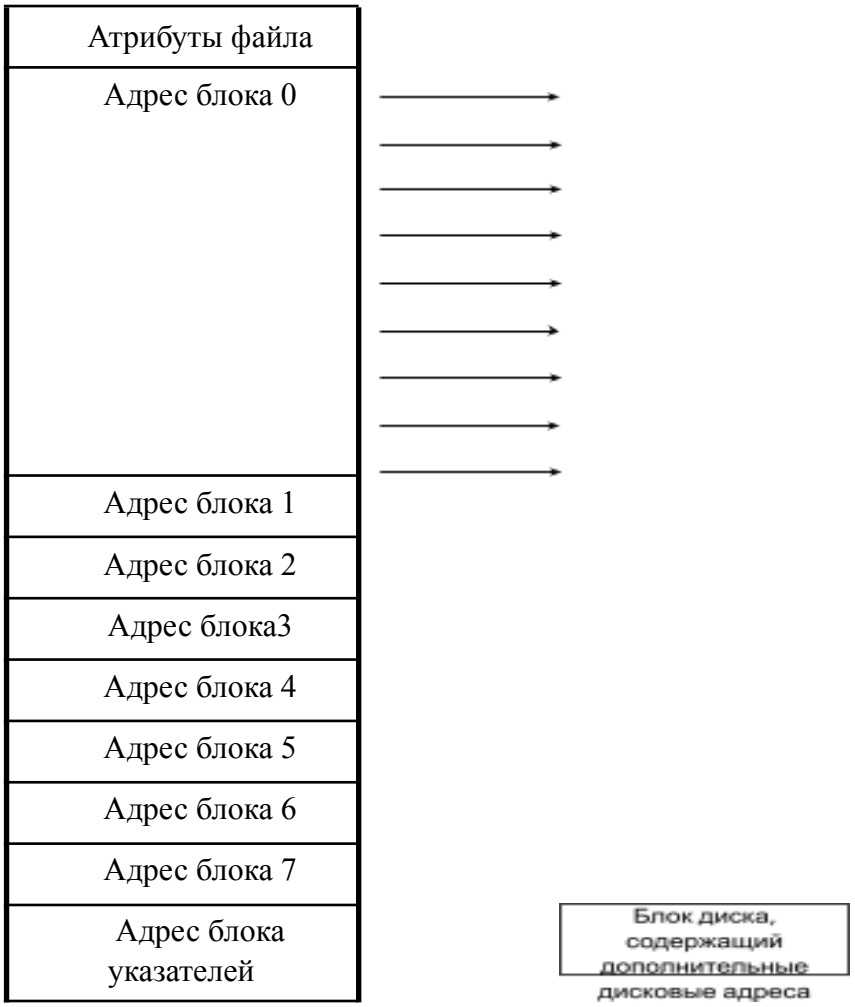


Рис. 5. Пример *i-того* узла

Корневой каталог обозначается символом «/», путь по дереву каталогов состоит из имен каталогов, разделенных символом «/», например: */usrlinclude/sys*. В каждый момент времени с любым пользователем связан текущий каталог, то есть местоположение пользователя в иерархической файловой системе. Каталог *UNIX* содержит по одной записи для каждого файла этого каталога. Каждая каталоговая запись проста, так как в системе *UNIX* используется схема *i-узлов* (рис.5.). Каталоговая запись состоит всего из двух полей: имени файла (14 байт) и номера *i-узла* для этого файла (рис.6.).

Каждый файл операционной системы *UNIX* может быть однозначно определен некоторой структурой данных, называемой *описателем файла* (*дескриптором*). Он содержит всю информацию о файле: тип файла, режим доступа, идентификатор владельца, размер, адрес файла, даты последнего доступа и последней модификации, дату создания и пр.

Обращение к файлу происходит по имени. Путь к файлу от корневого каталога называется *полным именем файла*. Если обращение к файлу начинается с символа «/», то считается, что указано полное имя файла и его поиск начинается с корневого каталога, в любом другом случае поиск файла начинается с текущего каталога. У любого файла может быть несколько имен. Фактически имя файла является ссылкой на файл, специфицированный номером описателя.

Байты: 2 14

Номер i-узла	Имя файла

Рис.6. Каталоговая запись файловой системы *UNUX V7*

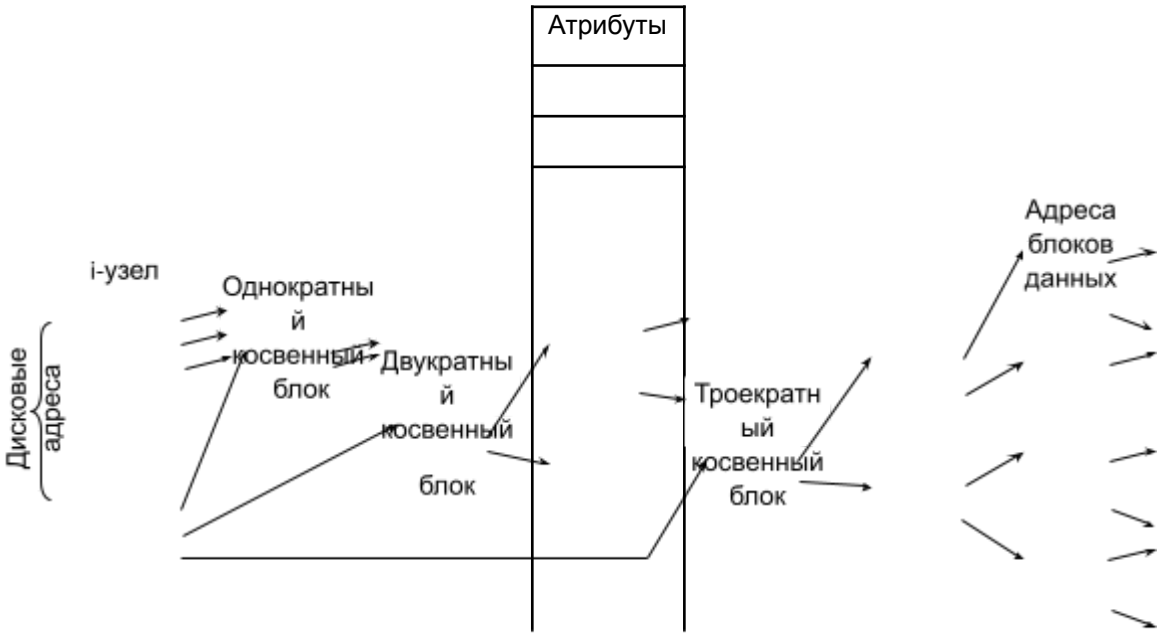
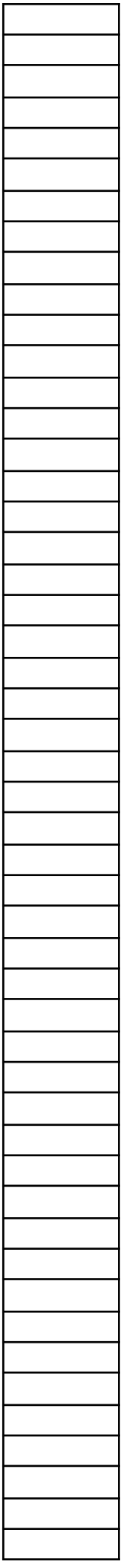


Рис.7. *I-тый* узел файловой системы *UNUX V7*



I-узлы системы *UNIX* (рис.7.) содержат атрибуты: размер файла, три указателя времени (создания, последнего доступа и последнего изменения), идентификатор владельца, номер группы, информацию о защите и счетчик каталоговых записей, указывающих на данный *i*-узел. При добавлении новой связи к *i*-узлу счетчик в *i*-узле увеличивается на единицу. При удалении связи счетчик в *i*-узле уменьшается на единицу. Когда значение счетчика достигает нуля, *i*-узел освобождается, а блоки диска, которые занимал файл, возвращаются в список свободных блоков.

Для учета дисковых блоков файла используется обобщение схемы, позволяющее работать с очень большими файлами. Первые 10 дисковых адресов хранятся в самом *i*-узле. Для небольших файлов необходимая информация содержится прямо в *i*-узле, считываемом с диска при открытии файла. Для файлов большего размера один из адресов в *i*-узле представляет собой адрес блока диска, называемого одинарным косвенным блоком. Этот блок содержит дополнительные дисковые адреса. Если и этого недостаточно используется другой адрес в *i*-узле, называемый двойным косвенным блоком и содержащий адрес блока, в котором хранятся адреса однократных косвенных блоков. Если и этого мало, используется тройной косвенный блок.

При открытии файла файловая система по имени файла находит его блоки на диске. Вначале файловая система открывает корневой каталог, *i*-узел которого располагается в фиксированном месте диска. По этому *i*-узлу система определяет положение корневого каталога, который может находиться в любом месте диска, например, в блоке 1. Затем файловая система считывает корневой каталог и ищет в нем первый компонент пути, чтобы определить номер *i*-узла файла. По этому *i*-узлу файловая система находит следующий каталог и находит в нем следующий компонент и т.д. Относительные пути файлов обрабатываются также как и абсолютные, с той разницей, что алгоритм начинает работу не с корневого, а с рабочего каталога.

Работа пользователя в системе начинается с того, что активизируется сервер терминального доступа *getty* (программа, организующая диалог работы с пользователем в многопользовательской операционной системе), который запускает программу *login*, запрашивающую у пользователя имя и пароль.

Далее происходит проверка *аутентичности* пользователя (подлинности регистрации пользователя в системе) в соответствии с той информацией, которая хранится в файле */etc/passwd*. В этом файле хранятся записи, содержащие:

- регистрационное имя пользователя;
- зашифрованный пароль;
- идентификатор пользователя;
- идентификатор группы;
- информация о минимальном сроке действия пароля;
- общая информация о пользователе;
- начальный каталог пользователя;
- регистрационный *shell* пользователя.

Если пользователь зарегистрирован в системе и ввел правильный пароль, *login* запускает программу, указанную в /etc/passwd – регистрационный *shell* пользователя.

Пользователь системы – это объект, обладающий определенными правами, определяющими возможность запуска программ на выполнение, а также владение файлами. Единственный пользователь системы, обладающий неограниченными правами – это *суперпользователь* или *администратор* системы.

Система идентифицирует пользователей по идентификатору пользователя (*UID – User Identifier*). Каждый пользователь является членом одной или нескольких групп – списка пользователей, имеющих сходные задачи. Каждая группа имеет свой уникальный идентификатор группы (*GID – Group Identifier*) Принадлежность группе определяет совокупность прав, которыми обладают члены данной группы.

Права пользователя UNIX – это права на работу с файлами. Файлы имеют двух владельцев: пользователя (*user owner*) и группу (*group owner*). Соответственно, атрибуты защиты файлов определяют права пользователя-владельца файла, права члена группы-владельца (g) и права всех остальных (o). В каждый момент времени с любым пользователем связан текущий каталог, то есть местоположение пользователя в иерархической файловой системе.

Всякий файл операционной системы *UNIX* в соответствии с его типом может быть отнесен к одной из следующих групп: обычные файлы, каталоги, специальные файлы и каналы.

Обычный файл представляет собой последовательность байтов. Никаких ограничений на файл системой не накладывается, и никакого смысла не приписывается его содержимому: смысл байтов зависит исключительно от программ, обрабатывающих файл.

Каталог – это файл особого типа, отличающийся от обычного файла наличием структуры и ограничением по записи: осуществить запись в каталог может только ядро операционной системы *UNIX*. Каталог устанавливает соответствие между файлами (номерами описателей) и их локальными именами.

Специальный файл – это файл, поставленный в соответствие некоторому внешнему устройству и имеющий специальную структуру. Его нельзя использовать для хранения данных как обычный файл или каталог, но над ним можно производить те же операции, что и над любым другим. При этом ввод/вывод информации в этот файл будет соответствовать вводу с внешнего устройства или выводу на него.

Канал – это программное средство, связывающее процессы операционной системы *UNIX* буфером ввода/вывода. Например, запуск процессов в виде *$процесс_1|процесс_2* означает, что стандартный вывод процесса_1 будет замкнут на стандартный ввод процесса_2. При этом сначала создается канал, а потом на выполнение одновременно запускаются оба процесса, и общее время их выполнения определяется более медленным процессом.

Файловые системы операционной системы Linux

В *Linux* долгое время была одна файловая система *Ext2fs* – вторая расширенная файловая система. Система определяется как расширенная по сравнению с файловой системой операционной системы *Minix*, послужившей прототипом *Linux* (до сих пор используемой на отформатированных в этой

операционной системе дискетах). Вторая – означает, что ранние версии *Linux* базировались на *Extfs* с более ограниченными возможностями.

По способу организации хранения данных *Extfs* напоминает файловую систему *Unix*. Отличительные особенности:

- дробление дискового раздела на группы блоков;
- наличие нескольких копий суперблока, что повышает надежность хранения данных;
- наличие эффективного механизма кэширования дисковых операций, что обеспечивает их быстродействие;
- относительно слабая устойчивость при аварийном завершении работы (вследствие мертвого зависания или отказа питания).

Проблема нарушения целостности файловой системы при некорректном завершении работы характерна и для всех операционных систем семейства *Unix*. Потому для реализации механизма восстановления стали разрабатывать *журналируемые файловые системы*. Журнал представляет собой *log*-файл дисковых операций, в котором фиксируются не выполненные, а только предстоящие манипуляции с файлами. Журналирование направленно на обеспечение целостности файловой системы, но не гарантирует сохранности пользовательских данных.

В большинстве журналируемых файловых систем фиксируются будущие операции только над метаданными изменяемых файлов, что достаточно для сохранения целостности файловой системы и предотвращения долговременных проверок, но не предотвращает потери данных в аварийных ситуациях. В некоторых файловых системах журналирование распространяется и на область данных файла, однако повышение надежности снижает быстродействие системы.

Текущие версии ядра *Linux* поддерживают в качестве альтернативных четыре журналируемые файловые системы: *ReiserFS*, *Ext3fs* и *XFS*, *JFS* (результаты импортирования в *Linux* файловых систем, разработанных первоначально для рабочих станций под операционные системы *Irix* (*SGI*) и *AIX* (*IBM*), соответственно).

Журналируемая файловая система *ReiserFS* разработана специально для *Linux* фирмой *Namesys* (*http://www.namesys.com*) и поддерживается ее ядром (*http://www.kernel.org*), начиная с первых версий ветви *2.4.x*. В *ReiserFS* осуществляется журналирование только операций над метаданными файлов, что при определенном снижении надежности, обеспечивает высокую производительность. Кроме этого, *ReiserFS* обладает уникальной (и по умолчанию задействованной) возможностью оптимизации дискового пространства, занимаемого мелкими, менее одного блока, файлами. Они хранятся в своих *inode*, без выделения блоков в области данных. Вместе с экономией места это способствует росту производительности, так как данные и метаданные (в терминах *ReiserFS – stat -data*) файла хранятся в непосредственной близости и могут быть считаны одной операцией ввода/вывода.

Хвосты файлов (их конечные части), меньшие по размеру, чем один блок, могут быть подвергнуты упаковке. Этот режим (*tailing*) включается по умолчанию при создании *ReiserFS*, обеспечивая около 5% экономии дискового пространства, но несколько снижает быстродействие. *ReiserFS* не совместима с *Ext2fs* на уровне утилит обслуживания файловой системы, но соответствующий инструментарий, разрешающий проблему, объединен в пакет *reiserfsprogs* и включен в штатный комплект современных дистрибутивов. Распространенные загрузчики *Linux* иногда не способны загрузить ядро *Linux* с раздела *ReiserFS*, поэтому *ReiserFS* не рекомендуется к употреблению на загрузочном разделе.

Ext3fs – представляет собой журналируемую надстройку над классической *Ext2fs*, разработанной в компании *Red Hat* и поддерживаемой ядром *Linux*, начиная с версии 2.4.16. Она сохраняет со своей прародительницей полную совместимость, в том числе и на уровне утилит обслуживания (начиная с версии 1.21, пакета *e2fsprogs*). Переход от *Ext2fs* к *Ext3fs* осуществляется добавлением файла журнала без переформатирования раздела и рестарта машины. *Ext3fs* является системой, в которой возможно журналирование операций не только с метаданными, но и с данными файлов, так как предусмотрено

три режима работы: полное журналирование (*full data journaling*); журналирование с обратной записью (*writeback*); последовательное журналирование, задействуемое по умолчанию (*ordered*).

Режим *полного журналирования* распространяется на метаданные и на данные файлов. Все их изменения сначала пишутся в файл журнала и только после этого фиксируются на диске. В случае аварийного отказа журнал можно повторно перечитать, приведя данные и метаданные в непротиворечивое состояние. Данный механизм гарантирует от потерь данных, однако является медленным.

В режиме *отложенной записи* в файл журнала записываются только изменения метаданных файлов и нет гарантии сохранности данных, однако обеспечивается наибольшее быстродействие. В *последовательном режиме* также физически журналируются только метаданные файлов, но связанные с ними блоки данных логически группируются в единый модуль (*транзакцию*, *transaction*) и записываются перед записью на диск новых метаданных, что способствует сохранности данных. Данные режим при меньших накладных расходах по сравнению с полным журналированием, обеспечивает промежуточный уровень быстродействия.

Файловая 64-разрядная система *XFS* развивается фирмой *SGI* для *Unix*, впервые появилась в версии *Irix 5.3*, вышедшей в 1994 г. *В Linux* она была импортирована недавно (*http://oss.sgi.com/projects/xfs*) и штатно поддерживается ядром, начиная с его ветки *2.6.X*. *XFS* представляет собой сбалансированную файловую систему, ориентированную на размещение в больших дисковых разделах для работы с большими файлами. Особенности *XFS*:

- использование механизма деления единого дискового раздела на несколько равных областей (*allocation group*), имеющих собственные списки *inodes* и свободные блоки, для распараллеливания дисковых операций (самостоятельные файловые субсистемы);
- использование логического журналирования только для изменений метаданных, но с частым сбросом их на диск для минимизации возможных потерь при сбоях;
- применение механизма ассигнования дискового пространства при записи файлов не во время журналирования, а при фактическом сбросе их на диск (*delayed allocation*), что вместе с повышением производительности предотвращает фрагментацию дискового раздела;
- использование списков контроля доступа (*ACL*, *Access Control List*) и расширенных атрибутов файлов (*extended attributes*).

Возможность работы с *XFS* обеспечивает специальный патч (*xfs-2.4.1X-all-i386.bz2*), который вместе с соответствующими утилитами поддержки можно получить с сайта SGI (*http://oss.sgi.com/projects/xfs*). Утилиты поддержки для *XFS* объединены в несколько пакетов.

Файловая система *JFS* разработана компанией *IBM* для собственной версии *Unix* и уже долгое время поддерживается ядром *Linux* в качестве альтернативной системы, однако по ряду причин широкого распространения она не получила.

На уровне обмена данными *Linux* поддерживает множество файловых систем, некоторые из них только в режиме чтения (например, *NTFS* или *HPFS*).

Все упомянутые выше файловые системы могут располагаться не только на реальных блочных устройствах (дисках и дисковых разделах), но и на виртуальных дисках (*RAM*-дисках).

Виртуальные файловые системы

Операционные системы *POSIX*-совместимые поддерживают еще и несколько типов виртуальных файловых систем, которые располагаются в оперативной памяти и служат для специальных целей. Первой виртуальной файловой системой была система процессов – *procfs*, которая представляла протекающие в системе процессы в виде файлов каталога */proc*, откуда получали информацию о процессах команды типа *ps* и *top*.

В каталоге */proc* каждому процессу соответствует подкаталог, именем которого является идентификатор процесса (номер в порядке запуска). Внутри такого подкаталога находится набор

файлов, содержимое некоторых из них можно посмотреть. Важная информация содержится в файлах корня /*proc*. Используя команды просмотра, можно посмотреть следующую информацию: о процессоре *(/proc/cpuinfo*); о текущей конфигурации ядра системы (/*proc/config.gz*); о загруженных его модулях *(/proc/modules*), об устройствах, подсоединенных к шине *PCI* (/*proc*/*pci*), сведения о сетевой карте и внутреннем модеме, необходимые для правильной их настройки.

Другой виртуальной файловой системой является файловая система устройств – *devfs*, которая по умолчанию задействована во *FreeBSD* 5-й ветки и во многих современных дистрибутивах *Linux*. Благодаря *devfs*, файлы устройств стали создаваться автоматически при старте системы (в соответствие с устройствами, поддерживаемыми текущей конфигурацией ядра и определяемыми в ходе загрузки операционной системы). Если в ходе работы недостанет какого-либо устройства, из числа поддерживаемых, то автоматически будет создан файл соответствующий этому устройству. Файловая система *devfs* сделала очень легким подключение устройств типа карт *PCMCIA*, *USB*-накопителей, цифровых камер и сканеров. Достаточно воткнуть *USB*-драйв в соответствующий разъем и в каталоге /*dev* появится соответствующий ему файл.

В настоящее время в *Linux* файловая система устройств постепенно заменяется системой поддержки динамического именования устройств – *udev*. В отличие от *devfs*, *udev* – пользовательская программа (входящая в одноименный пакет) и при ее использовании необходимость в поддержке *devfs* отпадает. Для своего функционирования *udev* нуждается в виртуальной файловой системе – *sysfs*, но ее поддержка в ядрах серии *3.6.X* осуществляется автоматически (а сама эта файловая система монтируется по умолчанию в каталог /*sys*). *Udev* присваивает имена всем устройствам, в том числе и при их подключении.

Любой *POSIX*-системе имена файлов конкретных устройств более или менее безразличны, так как оперирует она не с ними, а с их идентификаторами. Ранее в качестве таковых выступали старший и младший номера устройства, определяющие их класс и конкретный его экземпляр. В *udev* стали использоваться идентификаторы устройств (сериальный номер винчестера, его положение на канале *IDE*-шины и т.д.), сочетание которых для каждого диска (раздела) уникально. Программа *udev* извлекает эти сведения из файловой системы *sysfs* и, руководствуясь определенными правилами, ставит им в соответствие имена (например, /*dev*/*hda*).

Во *FreeBSD* и многих дистрибутивах *Linux* используется также файловая система в оперативной памяти, именуемая *mfs* в первом случае и *tmpfs* – во втором случае. Системы реализованы по-разному, но с точки зрения пользователя выглядят одинаково, как дисковые разделы, смонтированные в некоторые каталоги. Они заменяют собой блочные устройства там, где требуется быстрая, но не обязательно долговременная запись. Эти системы целесообразно использовать для промежуточных каталогов при архивации, разархивации, пакетной конвертации графических файлов, компиляции программ.

В отличие от стандартных файловых систем для хранения данных *mfs* и *tmpfs* не нуждаются в своем создании. Для их использования достаточно факта поддержки ядром (непосредственно или в виде модуля) и монтирования в какой-либо каталог. Обычно таким каталогом выступает /*tmp*, содержимое которого не должно сохраняться после рестарта системы. В *BSD*-системах в некоторых случаях целесообразно монтирование *mfs* в каталог /*usr*/*obj*, предназначенный для промежуточных продуктов компиляции при сборке ядра и базовых компонентов. В *Linux* одно из штатных мест монтирования *tmpfs* – каталог /*dev*/*sh*.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И ФОРМА ОТЧЕТНОСТИ:

Ответить на вопросы в тетради.

1. Что представляет собой файловая система *MS-DOS*?
2. Определите термин «аккаунт».
3. Определите термин «дескриптор файла».

4. Определите формат каталоговой записи в системе *MS-DOS*.
5. Определите формат каталоговой записи в системе *Windows 98*.
6. Определите вормат каталоговой записи с фрагментом длинного имени файла в *Windows 98*.
7. Каким образом реализуется отличие каталоговых записей, содержащих длинные и короткие имена файла в *Windows 98*?
8. Определите тип и назначение операционной системы *UNIX*.
9. Какую файловую структуру имеет операционная система *UNIX*?
10. Что фактически представляет имя файла в операционной системе *UNIX*?
11. С чего начинается работа пользователя в операционной системе *UNIX*?
12. Определите термин «аутентичность пользователя».
13. В соответствии с какой информацией реализуется аутентичность пользователя?
14. Перечислите права пользователя *UNIX*?
15. К каким группам относятся файлы операционной системы *UNIX*? Дать определение каждой из групп.
16. Перечислите файловые системы, поддерживаемые операционной системой *Linux*.
17. Определите отличие файловой системы *Ext2fs* от файловой системы *Ext3fs* и *ReiserFS*.
18. С помощью какого механизма в *Linux* и *UNIX* решается проблема защиты целостности файловой системы при некорректном завершении работы?
19. Перечислите особенности файловой системы *XFS*.
20. Определите назначение виртуальных файловых систем.
21. Перечислите виртуальные файловые системы, их назначение и отличие.

Литература:

1. Компьютерные сети: учеб. пособие/ А.В. Кузин, Д.А. Кузин. – 4-е изд., перераб. И доп. –М.: ФОРУМ: ИНФРА-М, 2018. – 190с.- (Среднее профессиональное образование) http://znanium.com/bookread2.php&book=938938
2. Компьютерные сети: учеб. пособие/ Н.В. Максимов, И.И. Попов. -6-е изд., перераб. И доп. –М: ФОРУМ: ИНФРА-М, 2019. -464 с. – (Среднее профессиональное образование) http://znanium.com/bookrea2.php&book=983166