**Цель**

Изучить особенности файловой системы FreeBSD, освоить работу с основными командами манипулирования файлами и каталогами, научиться монтировать файловые системы.

**Теоретические сведения**

Файловая система FreeBSD является ключевым моментом в понимании устройства всей системы. Самым важным понятием является, корневой каталог, обозначаемый символом **«/»**. Корневой каталог *монтиру­ется* самым первым на этапе загрузки и содержит все необходимое, чтобы подготовить систему к загрузке в многопользовательский режим. Корне­вой каталог также содержит точки монтирования всех других файловых систем.

**Точкой монтирования** – каталог, который будет соответствовать корню примонтированой файловой системы. Стандартные точки монтиро­вания включают /usr, /var, /tmp, /mnt и /cdrom. Эти каталоги обыч­но перечислены в файле **/etc/fstab**, в котором указаны файловые сис­темы и их точки монтирования.

|  |  |
| --- | --- |
| **Каталог** | **Описание** |
| / | Корневой каталог файловой системы. |
| /bin/ | Основные утилиты, необходимые для работы как в одно­пользовательском, так и в многопользовательском режимах. |
| /boot/ | Программы и конфигурационные файлы, необходимые для нормальной загрузки операционной системы. |
| /boot/defaults/ | Конфигурационные файлы с настройками по умолчанию, ис­пользуемые в процессе загрузки операционной системы. |
| /dev/ | Файлы устройств. |
| /etc/ | Основные конфигурационные файлы системы и скрипты. |
| /etc/defaults/ | Основные конфигурационные файлы системы с настройками по умолчанию. |
| /etc/mail/ | Конфигурационные файлы для систем обработки почты. |
| /etc/namedb/ | Конфигурационные файлы **ДЛЯ** службы DNS named. |
| /etc/periodic/ | Файлы сценариев, выполняемые ежедневно, еженедельно и ежемесячно. |
| /etc/ppp/ | Конфигурационные файлы для утилиты ppp. |
| /mnt/ | Пустой каталог, используемый как временная точка монти­рования. |
| /proc/ | Виртуальная файловая система, отображающая текущие про­цессы. |
| /root/ | Домашний каталог пользователя root. |
| /sbin/ | Системные утилиты и утилиты администрирования, необхо­димые для работы, как в однопользовательском, так и в мно­гопользовательском режимах. |
| /stand/ | Программы, необходимые для работы в автономном режиме (например, при установке системы). |
| /tmp/ | Временный каталог. |
| /usr/ | Большинство пользовательских утилит и приложений. |
| /usr/bin/ | Пользовательские утилиты и приложения общего назначе­ния. |
| /usr/include/ | Стандартные заголовочные файлы для языка С. |
| /usr/lib/ | Файлы стандартных библиотек. |
| /usr/libdata/ | Файлы данных для различных утилит. |
| /usr/libexec/ | Системные даемоны и утилиты (выполняемые другими про­граммами). |
| /usr/local/ | Локальные пользовательские приложения, библиотеки, и т.д. |
| /usr/obj/ | Архитектурно-зависимые файлы и каталоги, образующиеся в процессе сборки системы из исходных текстов в /usr/src. |
| /usr/sbin/ | Системные утилиты и утилиты администрирования (испол­няемые пользователем). |
| /usr/share/ | Архитектурно-независимые файлы. |
| /usr/src/ | Исходные тексты BSD и/или программ. |
| /usr/X11R6/ | Утилиты, приложения и библиотеки X11R6 (X Window System). |

**Организация дисков**

Наименьшая единица, которую FreeBSD использует для обращения к фай­лам, это имя файла. Имена файлов чувствительны к регистру, поэтому readme.txt и README.TXT — два разных файла. FreeBSD не использу­ет расширение файла (.txt) для определения программа это, документ или другой тип данных.

Файлы хранятся в каталогах. Каталоги могут не содержать файлов, или мо­гут содержать много сотен файлов. Каталоги также могут содержать дру­гие каталоги, что позволяет создавать иерархию каталогов один в другом. Это упрощает организацию данных.

Обращение к файлам происходит путем задания имени файла или каталога, дополняемого прямым слэшем /, за которым может следовать имя другого каталога. Если есть каталог *Каталог\_Первый*, содержащий каталог *Каталог\_Второй*, который со­держит файл *readme.txt*, полное имя, или *путь* к файлу будет *Каталог\_Первый*/*Каталог\_Второй*/*readme.txt*.

Каталоги хранятся в файловых системах. Каждая файловая система содер­жит один каталог на верхнем уровне, называемый *корневым каталогом* этой файловой системы. Этот корневой каталог может содержать другие каталоги.

FreeBSD не использует букв дисков, или других имен дисков в пути. Вместо этого, одна файловая система назначается *корневой файловой сис­темой.* Обращение к корневому каталогу корневой файловой системы происходит через /. Любая другая файловая система *монтируется* к кор­невой файловой системе. Неважно как много дисков есть в вашей системе FreeBSD, каждый каталог будет выглядеть как расположенный на том же диске.

**Преимущества нескольких файловых систем**

Различные файловые системы могут иметь различные *опции монтирова­ния.* Например, в целях безопасности корневая файловая система может быть смонтирована только для чтения, что делает невозможным случайное удаление или редактирование критически важного файла. Отделение фай­ловых систем, используемых пользователями для записи, таких как **/home**, от других файловых систем позволяет также монтировать их с параметром *nosuid;* этот параметр отменяет действие битов *suid/guid* на исполняемых файлах, в этой файловой системе, что потенциально повышает безопас­ность.

FreeBSD автоматически оптимизирует расположение файлов на файловой системе в зависимости от того, как файловая система используется. Фай­ловая система, содержащая множество мелких часто записываемых файлов, будет иметь оптимизацию, отличную от таковой для файловой системы, содержащей несколько больших файлов.

**Монтирование и размонтирование файловых систем**

Файловая система лучше всего представима в виде дерева, с корнем в /. Каталоги, /dev, /usr и прочие - это ветви дерева, которые, в свою оче­редь, являются корнями для поддеревьев, также имеющих ветви (/usr/local), и т.д.

Хорошей практикой является разнесение некоторых особо важных катало­гов на разные файловые системы. Например, /var, содержит log/, spool/, а также всевозможные временные файлы и нередко увеличивает­ся в размерах настолько, что занимает все свободное место на диске. По­этому лучше смонтировать /var отдельно, чтобы избежать переполнения корневой директории /.

**Команда mount**

Команда mount(8) используется для монтирования файловых систем:

**mount устройство точка\_монтирования,**

где **устройство** - имя монтируемого устройства;

**точка\_монтирования** - директория, в которую будет вмонтирована файловая система устройства.

Опции, которые может принимать команда mount:

* **-r** – монтировать файловую систему в режиме "только для чтения"
* **-w** – монтировать файловую систему в режиме "чтение-запись"
* **-t *fstype*** – монтировать файловую систему как систему указанного типа, по умолчанию, тип файловой системы – «ufs»

Для монтирования специальных файловых систем, например файловых систем Microsoft, существуют модификации команды mount, например:

**mount\_msdosfs** – для монтирования систем FAT16 и FAT32;

**mount\_ntfs** – для монтирования систем NTFS;

Для некоторых общеизвестных устройств команда mount принимает упрощенный вид, например, mount /cdrom, mount /mnt/floppy. Для других устройств, их имена необходимо указывать в явном виде, на­пример: /dev/ad0s**N** - для жестких дисков, /dev/da0s**N** - для USB Flash-карт, /dev/fd**N** - для приводов гибких дисков и т.д. Здесь **N** - порядковый номер устройства текущего типа. Посмотреть наличие устройств и их точные имена можно в директории /dev.

**Команда umount**

Команда umount(8) принимает в качестве параметра **точку монтирования** какой-либо файловой системы либо **имя устройства** либо опцию *-а* или *-A.* Кроме того, вы можете дополнительно указать опцию *-f* для форсирован­ного размонтирования файловой системы, и *-v* для получения более под­робной информации. Имейте ввиду, что это в общем случае опасно и по­тому не рекомендуется, так как тем самым вы можете нарушить работу компьютера или повредить данные на файловой системе. Опции *-а* и *-A* используются для размонтирования всех файловых систем (разве что вы укажете опцию *-t).* Разница состоит в том, что *-A* не пытается размонтировать корневую файловую систему.

**Команда df**

Позволяет посмотреть свободное место на дисках.

Утилита df выводит данные о размере свободного дискового пространства указанной файловой системы или файловой системы, к которой относится указанный файл. Выводимые значения соответствуют количеству 512-байтных блоков. сли не заданы ни файл, ни файловая система, утилита выводит статистику по всем cмонтированным файловым системам (зависит от опции -t, смотрите её описание ниже).

Имеются следующие опции:

* -a – Показывать все точки монтирования, включая те, что были смонтированы с флагом MNT\_IGNORE.
* -b – Использовать для вывода блоки по 512 байт.
* -c – Показывать общую сумму.
* -g – Использовать для вывода блоки по 1073741824 байт (1 Гбайт).
* -H – "Удобочитаемый" вывод. Использовать суффиксы B (байт), K (килобайт), M (мегабайт), G (гигабайт), T (терабайт) и P (петабайт) для уменьшения количества цифр в результате до 4-х или менее, используя для этого соответствующую степень числа 10.
* -h – "Удобочитаемый" вывод. Использовать суффиксы B (байт), K (килобайт), M (мегабайт), G (гигабайт), T (терабайт) и P (петабайт) для уменьшения количества цифр в результате до 4-х или менее, используя для этого соответствующую степень числа 2.
* -i – Дополнить отчёт количеством свободных индексных дескрипторов (inodes).
* -k – Использовать для вывода блоки по 1024 байт (1 Кбайт).
* -l – Отображать данные только о локально-смонтированных файловых системах.
* -m – Использовать для вывода блоки по 1048576 байт (1 Мбайт).
* -n – Показывать ранее полученную статистику по файловым системам.
* -P – Указывает на то, что вывод должен соответствовать стандарту POSIX. Выводимые значения будут в 512-байтных блоках.
* -t – Отображать данные только о файловых системах заданного типа. Можно задать несколько типов, разделив их список запятыми.

**Команды управления файловой системой**

Для управления файловой системой имеются различные команды, реализующие операции по созданию, чтению, копированию, переименованию/перемещению, изменению и удалению файлов и каталогов. Как правило, это специализированные команды, хорошо выполняющие свою задачу, однако некоторые функции могут частично дублироваться другими командами, что только добавляет гибкости управлению файлами.

Основными командами для выполнения файловых операций являются: **pwd, ls, cp, mv, rm, cd, rmdir, mkdir, ln**.

**Команда cp**

Позволяет копировать файлы.

Синтаксис:

cp [-R [-H | -L | -P]] [-f | -i | -n] [-lpv] исходный\_файл целевой\_файл

cp [-R [-H | -L | -P]] [-f | -i | -n] [-lpv] исходный\_файл целевой\_каталог

В первой форме утилита **cp** копирует содержимое файла исходный\_файл в целевой\_файл. Во второй форме содержимое каждого указанного файла копируется в целевой\_каталог. Имена самих файлов при этом не изменяются. Если **cp** обнаруживает попытку копирования файла самого в себя, копирование не производится.

Имеются следующие опции:

* -H –Если указана опция -R, следовать символическим ссылкам, указанным в командной строке, но не следовать символическим ссылкам, встречающимся в процессе обхода дерева каталогов.
* -L – Если указана опция -R, следовать всем символическим ссылкам.
* -P – Если указана опция -R, не следовать никаким символическим ссылкам. Это режим работы по умолчанию.
* -R – Если исходный\_файл указывает на каталог, **cp** копирует этот каталог вместе со всем деревом файловой иерархии, которое он содержит. Если исходный\_файл оканчивается на /, копируется не сам каталог, а только его содержимое. Эта опция также указывает на необходимость копирования символических ссылок, а не объектов, на которые они указывают, и заставляет **cp** создавать специальные файлы вместо того, чтобы копировать их, как обычные. Создаваемые каталоги имеют те же права доступа, что и исходные каталоги, вне зависимости от действующего для процесса значения umask. Обратите внимание, что **cp** копирует жёсткие ссылки как отдельные файлы.
* -f – Для каждого уже существующего целевого имени пути, удалять его и создавать новый файл, не требуя подтверждения операции, вне зависимости от его прав доступа.
* -i – Заставляет **cp** выдавать запрос через стандартный файл ошибок перед копированием файла, которое бы вызвало перезапись существующего файла. Если ответ, полученный со стандартного ввода, начинается символом `y' или `Y', будет предпринята попытка копирования.
* -l – Создавать жёсткие ссылки на обычные файлы в иерархии вместо того, чтобы их копировать.
* -n – Не перезаписывать существующие файлы.
* -p – Заставляет **cp** сохранять следующие атрибуты каждого исходного файла в создаваемой копии: время модификации, время доступа, флаги файла, режим доступа, идентификаторы пользователя и группы, насколько это позволено правами доступа. Если значения идентификаторов пользователя и группы не могут быть сохранены, не выводится никакого сообщения об ошибке, и возвращаемое программой значение статуса завершения не изменяется.
* -v – Выводить больше информации, показывая файлы по мере их копирования. Для каждого уже существующего целевого файла, его содержимое перезаписывается, если позволяют права. Его режим доступа и идентификаторы пользователя и группы не изменяются, если не задана опция -p.

**Команда mkdir**

Позволяет создать каталоги

Синтаксис

mkdir [-pv] [-m режим] имя\_каталога

Имеются следующие опции:

* -m – Установить права доступа конечного создаваемого каталога в соответствии с указанным режимом. Аргумент режим может быть задан в любом из форматов, определённых для команды chmod(1). Если используется символьный режим, то символы `+' и `-' интерпретируются относительно начального режима ``a=rwx''.
* -p – Создавать промежуточные каталоги по мере необходимости. Если эта опция не указана, то префикс полного пути каждого из операндов должен уже существовать. С другой стороны, если эта опция указана, то существование каталога, заданного операндом, не является ошибкой.
* -v – Выводить подробную информацию при создании каталогов, показывая их по мере создания. Пользователь должен иметь право на запись в родительский каталог.

**Команда mv**

Позволяет переместить файлы

Синтаксис

mv [-f | -i | -n] [-v] источник цель

mv [-f | -i | -n] [-v] источник каталог

В первом варианте вызова утилита mv переименовывает файл, заданный аргументом источник, в целевой путь, заданный аргументом цель. Такая форма подразумевается, когда последний операнд не является именем уже существующего каталога.

Во втором варианте mv перемещает каждый файл источник в целевой файл в существующем каталоге, заданным операндом каталог. Целевой путь для каждого операнда получается конкатенацией последнего операнда, косой черты, и заключительной части имени пути заданного файла.

Имеются следующие опции:

* -f – Не запрашивать подтверждение перед перезаписью целевого пути.
* -i – Заставляет mv выдавать запрос через стандартный файл ошибок перед перемещением файла, которое бы вызвало перезапись существующего файла. Если ответ со стандартного ввода начинается символом `y' или `Y', то будет сделана попытка выполнить перемещение.
* -n – Не перезаписывать существующие файлы.
* -v – Выводить больше информации, показывая файлы по мере их перемещения.

Указание каталога в качестве операнда источник является ошибкой, если цель существует, и не является каталогом.

Если права доступа к целевому пути не позволяют выполнить запись, то mv запросит подтверждение у пользователя, как если бы была указана опция -i.

Поскольку системный вызов rename(2) работает только в пределах одной файловой системы, mv использует cp(1) и rm(1) для выполнения перемещения.

Эффект аналогичен выполнению последовательности команд:

rm -f целевой\_путь && \

cp -pRP файл\_источник цель && \

rm -rf файл\_источник

**Команда rm**

rm, unlink – удалить элементы каталога

Синтаксис:

rm [-f | -i] [-dIPRrvW] файл ...

unlink файл

Утилита **rm** пытается удалить файлы, указанные в командной строке и не являющиеся каталогами. Если права доступа к файлу не разрешают запись и устройством стандартного ввода является терминал, то пользователю выводится запрос (через стандартный файл ошибок) на подтверждение действия.

Имеются следующие опции:

* -d – Пытаться удалять каталоги так же, как и файлы других типов.
* -f – Пытаться удалять файлы без запроса подтверждения, вне зависимости от установленных прав доступа к файлу. Если файл не существует, не выводить никакой диагностической информации и не менять статус завершения для информирования об ошибке.
* -i – Запрашивать подтверждение перед удалением каждого файла, несмотря на установленные права доступа к файлу, и вне зависимости от того, является ли устройство стандартного ввода терминалом или нет.
* -I – Запрашивать подтверждение только если удалению подлежит более трёх файлов, или перед рекурсивным удалением каталога. Установка этой опции позволяет существенно снизить количество запросов (по сравнению с опцией -i), и в то же время обеспечивает практически такой же уровень защиты от ошибок.
* -P – Перезаписывать обычные файлы перед удалением. Файлы перезаписываются трижды, сначала значением 0xff, затем 0x00, затем ещё раз 0xff, и только потом удаляются. Файлы с несколькими жёсткими связями не будут ни перезаписаны, ни удалены (если только не указана опция -f), а вместо этого будет выводиться предупреждение. Указание этого флага для файла, имеющего доступ только на чтение, приведёт к выводу сообщения об ошибке и завершению работы rm. Файл при этом не будет перезаписан или удалён.
* -R – Пытаться удалить дерево каталогов, корень которого указан аргументом файл.
* -r – То же самое, что и -R.
* -v – Выводить больше информации при удалении файлов, показывая их по мере их удаления.
* -W – Пытаться восстановить перечисленные файлы.

Утилита **rm** удаляет символические ссылки, а не сами файлы, на которые эти ссылки указывают. Попытка удалить файлы **/, .** или **..** вызовет ошибку.

**Команда rmdir**

Позволяет удалить каталоги

Синтаксис

rmdir [-pv] каталог

Утилита **rmdir** удаляет каталоги, заданные аргументами каталог, в том случае, если они пусты.

Аргументы обрабатываются в указанном порядке, поэтому для удаления и родительского каталога, и его подкаталога вначале должен быть указан подкаталог, чтобы родительский каталог был уже пуст на момент, когда **rmdir** попытается его удалить.

Имеются следующие опции:

* -p – Каждый каталог рассматривается как путь, все компоненты которого будут удалены, если они пусты, начиная с последнего компонента.
* -v – Выводить подробную информацию, показывая каталоги по мере их удаления.

**Команда ln**

Позволяет создать ссылки

Синтаксис:

ln [-Ffhinsv] исходный\_файл [целевой\_файл]

ln [-Ffhinsv] исходный\_файл ... целевой\_каталог

link исходный\_файл целевой\_файл

Утилита **ln** создаёт в каталоге новую запись (ссылку на файл), все характеристики которой совпадают с исходным файлом. Это позволяет иметь одновременно несколько копий файла в разных местах, не тратя при этом место на диске для копий; вместо этого ссылка «указывает» на оригинал. Существует два типа ссылок: **жёсткие** и **символические**. Каким образом ссылка «указывает» на файл, зависит от типа ссылки.

Имеются следующие опции:

* -f – Если целевой файл уже существует, удалить его, чтобы можно было создать ссылку.
* -F – Если целевой файл уже существует и является каталогом, удалить его, чтобы можно было создать ссылку. Опцию -F следует использовать с опциями -f или -i, и если ни одна из них не указана, то подразумевается опция -f. Опция -F ничего не делает, если не указана опция -s.
* -h – Если целевой\_файл или целевой\_каталог является символической ссылкой, не следовать ей. Эта опция особенно полезна в сочетании с опцией -f для замены символической ссылки, которая может указывать на каталог.
* -i – Заставляет ln выдавать запрос через стандартный файл ошибок в случае, если целевой файл существует. Если ответ, полученный со стандартного ввода, начинается символом `y' или `Y', удалить целевой файл, чтобы можно было создать ссылку. В противном случае, не пытаться создавать ссылку.
* -n – То же, что и -h, для совместимости с другими реализациями ln.
* -s – Создать символическую ссылку.
* -v – Выводить больше информации, показывая обрабатываемые утилитой файлы.

По умолчанию **ln** создаёт жёсткие ссылки. Жёсткая ссылка на файл ничем не отличается от исходного файла; при этом любые изменения в файле совершенно не зависят от имени, по которому к нему обратились. Жёсткие ссылки не могут ссылаться на каталоги, и не могут находиться на другой файловой системе.

Символическая ссылка содержит имя файла, на который она ссылается. При выполнении операции open(2) над символической ссылкой используется оригинальный файл. Вызов stat(2), выполненный над символической ссылкой, также вернёт исходный файл. Для получения информации о ссылке следует использовать lstat(2). Для чтения содержимого символической ссылки можно воспользоваться вызовом readlink(2). Символические ссылки могут находиться на другой файловой системе и могут указывать на каталоги.

Важно понимать разницу между «жесткими» и «мягкими» ссылками.

|  |  |
| --- | --- |
| **«Жесткие» ссылки** | **«Мягкие» ссылки** |
| Возможны только в рамках одной физической файловой системы (в рамках одного дискового раздела) | Возможны на любые объекты в рамках файловой системы (даже между дисковыми разделами) |
| Указывают на саму область данных в файловой системе | Указывает не на сами данные, а на символьное имя файла внутри файловой системы, т.е. на одну существующую «жесткую» ссылку или на другую «мягкую» ссылку |
| Все «жесткие» ссылки на один файл первичны и равнозначны | Является вторичной ссылкой по отношению к объекту файловой системы, на символьное имя которого она указывает |
| При удалении одной из нескольких «жестких» ссылок на один и тот же файл сам файл не удаляется, все остальные «жесткие» ссылки продолжают существовать и обеспечивать к нему доступ. Сам файл перестает существовать только после удаления последней «жесткой» ссылки на него.  При первичном создании файла счетчик его «жестких» ссылок устанавливается в «1». При добавлении «жесткой» ссылки на тот же файл счетчик увеличивается, при удалении счетчик уменьшается. При обнулении счетчика данные помечаются удаленными. | При удалении «мягкой» ссылки сам файл продолжает существовать.  При удалении «жесткой» ссылки, на символьное имя которой указывает «мягкая» ссылка, эта «мягкая» ссылка продолжает существовать, но становиться «не рабочей» и не обеспечивает доступ к данным (даже если сами данные не удалены и доступны по другим «жестким» ссылкам). |

Рассмотрим пример. Создадим файл file\_a с содержимым «aaa»:

**echo "aaa" > file\_a**

Создадим на него 2 ссылки: символическую и жёсткую:

**ln -s file\_a link1**

**ln file\_a link2**

Попробуем достучаться до файла по обеим ссылкам:

**cat link1**

получим - **aaa**

**cat link2**

получим - **aaa**

Теперь удалим оригинальный файл и попробуем достучаться ещё раз. Первая ссылка окажется «битой», а по второй ссылке по-прежнему находится файл с содержимым «aaa»:

**rm file\_a**

**ls**

получаем - **link1 link2**

**cat link1**

**cat: link1**

получаем - **cat: link1: No such file or directory**

**cat link2**

получаем - **aaa**

Снова создадим файл с именем file\_a, но с другим содержимым:

**echo "bbb" > file\_a**

А теперь снова попробуем достучаться по ссылкам. Первая ссылка снова работает, и по ней будет находиться файл с содержимым «bbb». А по второй ссылке находится всё тот же файл с содержимым «aaa».

**cat link1**

получаем - **bbb**

**cat link2**

получаем - **aaa**

**Задание**

1. Войдите в систему с учетной записью **Student**.
2. Выведите на экран список файлов текущего каталога в краткой и расширенной форме.
3. Сделать каталог **/** текущим.
4. Сохранить в файле **$HOME/filelist.txt** список каталогов в каталоге /. **($HOME** - переменная окружения, которая содержит путь к «домашнему» каталогу пользователя).
5. Вернитесь в домашний каталог и выведите рекурсивный список всех файлов и каталогов в каталоге **/usr/ports/www**.
6. В домашнем каталоге создайте подкаталоги **src**, **dst**, **temp**.
7. В каталоге **src** создайте текстовый файл **f1** произвольного содержания.
8. В каталог **src** скопируйте файлы **info.txt**, **ps.txt** и **ps1.txt**, созданные в предыдущей работе. Все ли файла получилось скопировать?
9. В каталоге **dst** создайте «жесткие» ссылки на все файлы из каталоге **src.**
10. В домашнем каталоге создайте «мягкие» ссылки на файлы из каталоге **src**.
11. Выведите рекурсивно расширенную информацию о содержимом домашнего каталога. Обратите внимание на размер для физических файлов и ссылок. Результат сохраните в файл в домашнем каталоге.
12. Находясь в домашнем каталоге выполнить команды и запомните результат:

* **Cat src/f1**
* **Cat dst/f1\_hard\_link.txt**
* **Cat f1\_soft\_link.txt**

1. Переместите файл **f1** из каталога **src** в каталог **temp** и повторите пункт 12.
2. Удалите файл **f1** и повторите пункт 12.
3. Удалите все файлы с расширением **.txt** из каталога **dst**.
4. Удалите каталог **dst.**
5. Переместите каталог **temp** в **src.**
6. Выведите рекурсивно расширенную информацию о содержимом домашнего каталога. Обратите внимание на размер для физических файлов и ссылок. Результат сохраните в файл в домашнем каталоге.
7. Рекурсивно удалите каталог **src.**
8. Завершите сеанс.
9. Войдите в систему с учетной записью **root**.
10. Выполните команду **df –h**. Обратите внимание на первый и последний столбцы результата. Первый – это наименование устройства, соответствующее дискам, а второй это точка монтирование файловой системы. Нас будет интересовать второй диск, на который мы при инсталляции установили порты.
11. С помощью команды **umount** отмонтируйте файловую систему второго (меньшего) диска.
12. Посмотрите результат с помощью команды **df**.
13. С помощью команды **mount** восстановите первоначальное состояние – примонтируйте диск в **/usr/ports**.
14. Завершите сеанс и выключите виртуальную машину.