ЛАБОРАТОРНАЯ РАБОТА №6 ФАЙЛОВЫЕ СИСТЕМЫ ОС LINUX

Цель работы – практическое знакомство с организацией данных основной файловой системы ОС Linux и используемыми утилитами.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Файл

Данные, хранящиеся на любом носителе, образуют *файл* Linux. Более того, многие устройства, подключенные к компьютеру (начиная с клавиатуры и заканчивая любыми внешними устройствами), Linux представляет как файлы (так называемые *специальные файлы*). В Linux определено несколько различных типов файлов. В основном пользователь имеет дело с файлами трех типов: обычными файлами, предназначенными для хранения данных, *каталогами* и *файлами-ссылками.*

Система файлов: каталоги

Файловая система имеет иерархическую структуру. Linux может работать с различными типами файловых систем. В этой работе будут описаны возможности файловой системы Ext3fs. В файловой системе Ext3fs каждый *каталог* - это отдельный *файл* особого типа ("d", от англ. "directory"), отличающийся от обычного файла с данными: в нем могут содержаться только ссылки на другие файлы и каталоги.

Допустимые имена файлов и каталогов

Linux всегда **различает** заглавные и строчные буквы в именах файлов и каталогов, поэтому "`student`", "`Student`" и "`STUDENT`" будут тремя **разными** именами.

Есть несколько символов, допустимых в именах *файлов* и *каталогов,* которые нужно использовать с осторожностью. Это *спецсимволы* "`*`", "`\`", "`&`", "`<`", "`>`", "`;`", "`(`", "`)`", "`|`", а также символы пробела и табуляции.

Кодировки и расширения

В Linux в именах *файлов* и *каталогов* допустимо использовать не только символы латинского алфавита, но и любые символы любого языка.

В файловой системе Linux нет никаких предписаний по поводу расширения: в *имени файла* может быть любое количество точек (в том числе ни одной), а после последней точки может стоять любое количество символов. Хотя расширения не обязательны, они широко используются: расширение позволяет программе, не открывая файл, только по его имени определить, какого типа данные в нем содержатся. Определить тип содержимого файла можно и на основании самих данных (сигнатур). Многие форматы предусматривают указание в начале файла, как следует интерпретировать дальнейшую информацию.

В Linux есть утилита `file`, которая предназначена для определения типа содержащихся в файле данных. Эта утилита никогда не доверяет расширению *файла* (если оно присутствует), а анализирует сами данные. `file` различает не только разные данные, но и разные типы *файлов*.

1.1 Дерево каталогов

В большинстве современных *файловых систем* используется иерархическая модель организации данных: существует один *каталог*, объединяющий все данные в *файловой системе* - это "корень" всей *файловой системы*, ***корневой каталог***. *Корневой каталог* может содержать любые объекты *файловой системы*, и в частности, *подкаталоги*. Подкаталоги также могут содержать любые объекты *файловой системы* и *подкаталоги* и т. д. Таким образом, **все**, что записано на диске - *файлы, каталоги* и специальные *файлы* - обязательно "принадлежит" *корневому каталогу*: либо непосредственно (содержится в нем), либо на некотором уровне вложенности.

Структуру *файловой системы* можно представить наглядно в виде дерева, "корнем" которого является *корневой каталог*, а в вершинах расположены все остальные *каталоги*. На рис. 1 изображено дерево *каталогов*, курсивом обозначены имена *файлов*, прямым начертанием - имена *каталогов*.

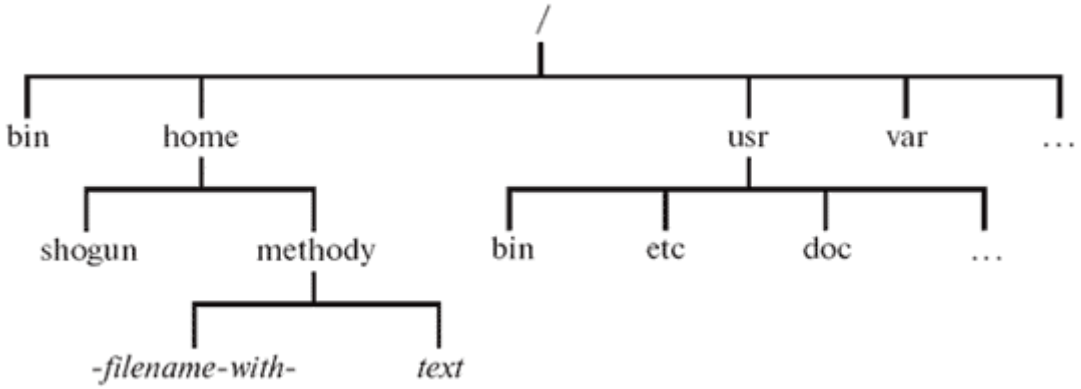


Рис. 1. Дерево каталогов в ext3fs

В любой *файловой системе* Linux всегда есть только один *корневой каталог*, который называется "/". Пользователь Linux всегда работает с единым деревом *каталогов*, даже если разные данные расположены на разных носителях: жестких или сетевых дисках, съемных дисках, CD-ROM и т. п. Такое представление отличается от технологии, применяемой в Windows, где для каждого устройства, на котором есть файловая система, используется свой корневой каталог, обозначенный литерой, например "a", "c", "d" и т. д. Для того чтобы отключать и подключать файловые системы на разных устройствах в состав одного общего дерева, используются процедуры монтирования и размонтирования. После того, как файловые

системы на разных носителях подключены к общему дереву, содержащиеся на них данные доступны так, как если бы все они составляли единую файловую систему: пользователь может даже не знать, на каком устройстве какие файлы хранятся.

Положение любого *каталога* в дереве *каталогов* описывается при помощи *полного пути*. *Полный путь* всегда начинается от корневого каталога и состоит из перечисления всех вершин, встретившихся при движении по ветвям дерева до искомого *каталога* включительно. Названия соседних вершин разделяются символом "/" ("слэш"). В Linux *полный путь*, например, до *каталога* "methody" в *файловой системе*, приведенной на рис. 1, записывается следующим образом /home/methody.

1.2 Размещение компонентов системы: стандарт FHS

Фрагмент дерева *каталогов* типичной файловой системы Linux приведен на рис. 1. Утилита ls выведет список всего, что в этом каталоге содержится.

Пример 1. Стандартные каталоги в / . Использование утилиты ls

```
[student@localhost ~]$ ls /
bin  dev  home  lost+fount  misc  net  proc  tmp  var
boot  etc  lib  sbin  usr
```

В примере 1 утилита ls вывела список подкаталогов корневого каталога. Этот список будет примерно таким же в любом дистрибутиве Linux. В *корневом каталоге* Linux-системы обычно находятся только подкаталоги со **стандартными** именами. Более того, не только имена, но и **тип данных**, которые могут попасть в тот или иной *каталог*, также регламентированы стандартом **Filesystem Hierarchy Standard** ("стандартная структура *файловых систем*"). Краткое описание стандартной иерархии каталогов Linux можно получить, выполнив команду man hier. Полный текст и последнюю редакцию стандарта FHS можно прочесть по адресу http://www.pathname.com/fhs/

Содержимое подкаталогов корневого *каталога*.

/bin Название этого *каталога* происходит от слова "binaries" ("двоичные", "исполняемые"). В этом *каталоге* находятся исполняемые *файлы* самых необходимых утилит, которые могут понадобиться системному администратору или другим пользователям.

/boot "Boot" - загрузка системы. В этом *каталоге* находятся *файлы*, необходимые для загрузки ядра - и, обычно, само *ядро*. Пользователю практически никогда не требуется непосредственно работать с этими *файлами*.

`/dev` В этом *каталоге* находятся все имеющиеся в системе *файлы* особого типа, предназначенные для обращения к различным системным ресурсам и устройствам. Например, *файлы* `/dev/ttyN` соответствуют *виртуальным консолям*, где `N` - номер *виртуальной консоли*. Данные, введенные пользователем на первой *виртуальной консоли*, система считывает из *файла* `/dev/tty1;` в этот же *файл* записываются данные, которые нужно вывести пользователю на эту консоль. В специаьных *файлах* в действительности не хранятся никакие данные, при их помощи данные **передаются.**

`/etc` *Каталог* для системных *конфигурационных файлов*. Здесь хранится информация о специфических настройках данной системы: информация о зарегистрированных пользователях, доступных ресурсах, настройках различных программ.

`/home` Здесь расположены *каталоги*, принадлежащие пользователям системы - ***домашние каталоги***, отсюда и название "home". Отделение всех *файлов*, создаваемых пользователями, от прочих системных *файлов* дает очевидное преимущество: серьезное повреждение системы или необходимость обновления не затронет пользовательских *файлов*.

`/lib` Название этого *каталога* - сокращение от "libraries" (англ. "библиотеки"). Чтобы не включать эти функции в текст каждой программы, используются стандартные функции библиотек - это значительно экономит место на диске и упрощает написание программ. В этом *каталоге* содержатся библиотеки, необходимые для работы наиболее важных системных утилит, размещенных в `/bin` и `/sbin`.

`/mnt` *Каталог* для *монтирования* (от англ. "mount") - временного подключения *файловых систем*, например, на съемных носителях (CD-ROM и др.).

`/proc` В этом *каталоге* все *файлы* "виртуальные" - они располагаются не на диске, а в оперативной памяти. В этих *файлах* содержится информация о программах (*процессах*), выполняемых в данный момент в системе.

`/root` *Домашний каталог* администратора системы - пользователя `root`. Смысл размещать его отдельно от *домашних каталогов* остальных пользователей состоит в том, что `/home` может располагаться на отдельном устройстве, которое не всегда доступно (например, на сетевом диске), а *домашний каталог* `root` должен присутствовать в любой ситуации.

`/sbin` *Каталог* для важнейших системных утилит (название *каталога* -

сокращение от "system binaries"): в дополнение к утилитам `/bin` здесь находятся программы, необходимые для загрузки, резервного копирования, восстановления системы. Полномочия на исполнение этих программ есть только у системного администратора.

`/tmp` Этот *каталог* предназначен для *временных файлов*: в таких *файлах* программы хранят необходимые для работы промежуточные данные. После завершения работы программы *временные файлы* теряют смысл и должны быть удалены. Обычно *каталог* `/tmp` очищается при каждой загрузке системы.

`/usr` Здесь можно найти такие же *подкаталоги* `bin`, `etc`, `lib`, `sbin`, как и в *корневом каталоге*. Однако в *корневой каталог* попадают только утилиты, **необходимые** для загрузки и восстановления системы в аварийной ситуации - **все остальные** программы и данные располагаются в *подкаталогах* `/usr`. Этот раздел *файловой системы* может быть очень большим.

`/var` Название этого *каталога* - сокращение от "variable" ("переменные" данные). Здесь размещаются те данные, которые создаются в *процессе* работы разными программами и предназначены для передачи другим программам и системам (очереди печати, электронной почты и др.) или для сведения системного администратора (системные журналы, содержащие протоколы работы системы). В отличие от каталога `/tmp` сюда попадают те данные, которые могут понадобиться после того, как создавшая их программа завершила работу.

Стандарт *FHS* регламентирует не только перечисленные каталоги, но и их подкаталоги, а иногда даже приводит список конкретных файлов, которые должны присутствовать в определенных каталогах. Этот стандарт последовательно соблюдается во всех Linux-системах.

Командная оболочка "знает", что исполняемые файлы располагаются в каталогах `/bin`, `/usr/bin` и т. д. - именно в этих каталогах она ищет исполняемый файл `cat`. Благодаря этому каждая вновь установленная в системе программа немедленно оказывается доступна пользователю из командной строки. Для этого не требуется ни перезагружать систему, ни запускать какие-либо процедуры - достаточно просто поместить исполняемый файл в один из соответствующих каталогов.

Рекомендации стандарта по размещению файлов и каталогов основываются на принципе размещения файлов, которые по-разному используются в системе, в разных подкаталогах. По типу использования файлы можно разделить на следующие группы:

пользовательские/системные файлы

Пользовательские файлы - это все файлы, созданные пользователем

и не принадлежащие ни одному из компонентов системы.

изменяющиеся/неизменные файлы

К неизмененным файлам относятся все статические компоненты программного обеспечения: библиотеки, исполняемые файлы и т. д. - все, что не изменяется само без вмешательства системного администратора. Изменяющиеся файлы изменяются без вмешательства человека в процессе работы системы: системные журналы, очереди печати и пр. Выделение неизменных файлов в отдельную структуру (например, `/usr`) позволяет использовать соответствующую часть файловой системы в режиме "только чтение", что уменьшает вероятность случайного повреждения данных и позволяет применять для хранения этой части файловой системы CD-ROM и другие носители, доступные только для чтения.

разделяемые/неразделяемые файлы

Это разграничение становится полезным, если речь идет о сети, в которой работает несколько компьютеров. Значительная часть информации при этом может храниться на одном из компьютеров и использоваться всеми остальными по сети (к такой информации относятся, например, многие программы и домашние каталоги пользователей). Однако часть файлов нельзя разделять между системами (например, файлы для начальной загрузки системы).

Полный путь к каталогу формально ничем не отличается от пути к файлу, т. е. по полному пути нельзя сказать наверняка, является его последний элемент файлом или каталогом. Чтобы отличать путь к каталогу, иногда используют запись с символом "/" в конце пути, например "/home/student/".

1.3 Текущий каталог

Каждая выполняемая программа "работает" в строго определенном *каталоге* файловой системы. Такой каталог называется *текущим каталогом***.** Можно представлять, что программа во время работы "находится" именно в этом каталоге, это ее "рабочее место". В зависимости от текущего каталога поведение программы может меняться: зачастую программа будет по умолчанию работать с файлами, расположенными именно в текущем каталоге - до них она "дотянется" в первую очередь. Текущий каталог есть у любой программы, в том числе и у командной оболочки (shell) пользователя. Поскольку пользователь взаимодействует с системой через командную оболочку, можно говорить о том, что пользователь "находится" в том *каталоге*, который в данный момент является текущим каталогом его командной оболочки.

Все команды, выполняемые пользователем при помощи shell, наследуют *текущий каталог* shell, т. е. "работают" в том же *каталоге*. По этой причине пользователю важно знать *текущий каталог* shell. Для этого

служит утилита pwd:

Команда pwd (print working directory) возвращает *полный путь* текущего каталога командной оболочки - естественно, именно той командной оболочки, при помощи которой была выполнена команда pwd.

Почти все утилиты по умолчанию читают и создают файлы в *текущем каталоге*. Например, утилита cat (concatenation – конкатенация) - выводит на экран содержимое файла "text":

[student@localhost student]$ cat text

В действительности, командная оболочка, прежде чем передавать параметр "text" (имя файла) утилите cat, подставляет значение *текущего каталога* - получается *полный путь* к этому файлу в *файловой системе*: "/home/student/text". Содержимое данного файла утилита cat выведет на экран.

Относительный путь(relative path) - путь к объекту *файловой системы*, не начинающийся в *корневом каталоге*. Для каждого *процесса* Linux определен *текущий каталог*, с которого система начинает *относительный путь* при выполнении файловых операций.

Относительный путь строится точно так же, как и полный - перечислением через "/" всех названий *каталогов*, встретившихся при движении к искомому *каталогу* или файлу. Между полным и *относительным путем* есть только одно существенное различие: *относительный путь* начинается **от текущего каталога**, в то время как *полный путь* всегда начинается **от корневого каталога**. *Относительный путь* любого файла или каталога в файловой системе может иметь любую конфигурацию - чтобы добраться до искомого файла, можно двигаться как по направлению к *корневому каталогу*, так и от него. Linux различает полный и *относительный пути* очень просто: если имя объекта начинается на "/" - это *полный путь*, в любом другом случае - относительный.

Отделить *путь* к файлу от его имени можно с помощью команд dirname и basename соответственно.

1.4 Домашний каталог

В Linux у каждого пользователя обязательно есть **собственный каталог**, который и становится текущим сразу после *регистрации в системе* - домашний каталог.

Домашний каталог (home directory) - это каталог, предназначенный для хранения собственных данных пользователя Linux. Как правило, является текущим непосредственно после регистрации пользователя в системе. *Полный путь* к *домашнему каталогу* хранится в *переменной окружения* HOME. Имя домашнего каталога ~

Поскольку каждый пользователь располагает собственным

каталогом и по умолчанию работает в нем, решается задача разделения файлов разных пользователей. Обычно доступ других пользователей к чужому *домашнему каталогу* ограничен: наиболее типична ситуация, когда пользователи могут читать содержимое файлов друг друга, но не имеют права их изменять или удалять.

1.5 Информация о содержимом каталога – утилита ls

Чтобы иметь возможность ориентироваться в *файловой системе*, нужно знать, что содержится в каждом *каталоге*. Просмотреть содержимое любого *каталога* можно при помощи утилиты ls (сокращение от англ. "list" - "список"):

Команда ls без параметров выводит список файлов и каталогов, содержащихся в текущем каталоге. Утилита ls принимает один параметр - имя каталога, содержимое которого нужно вывести. Имя может быть задано любым доступным способом: в виде полного или *относительного пути*.

Кроме параметра, утилита ls может использовать множество ключей, которые нужны для того, чтобы выводить дополнительную информацию о файлах в каталоге или выводить список файлов выборочно. Чтобы узнать обо всех возможностях ls, нужно прочесть *руководство* по этой утилите с помощью команды man ls.

Ключ –F используется, чтобы отличать файлы от каталогов. При наличии этого ключа ls в конце имени каждого каталога ставит символ "/", чтобы показать, что в нем может содержаться что-то еще.

Утилита ls по умолчанию не выводит информацию об объектах, чье имя начинается с "." - в том числе о "." и "..". Для того чтобы посмотреть полный список содержимого *каталога*, и используется ключ "-a" (all). Как правило, с "." начинаются имена *конфигурационных файлов* и *конфигурационных каталогов* (вроде .bashrc), работа с которыми (т. е. **настройка** окружения, "рабочего места") не пересекается с работой над какой-нибудь прикладной задачей.

Родительский каталог (parent directory) - это *каталог*, в котором содержится данный. Для корневого каталога родительским является он сам.

Ссылки на текущий и на *родительский каталог* обязательно присутствуют в **каждом каталоге** в Linux. Даже если каталог пуст, т. е. не содержит ни одного файла или подкаталога, команда "ls -a" выведет список из двух имен: "." и "..". За ссылками на текущий и *родительский каталоги* могут следовать несколько файлов и каталогов, имена которых начинаются с ".". В них содержатся настройки командной оболочки (файлы, начинающиеся с ".bash") и других программ. В *домашнем каталоге* каждого пользователя Linux всегда присутствует несколько таких файлов.

Использование этих файлов позволяет пользователям независимо друг от друга настраивать поведение командной оболочки и других программ - организовывать свое "рабочее место" в системе.

1.6 Перемещение по дереву каталогов – команда cd

Пользователь может работать с файлами не только в своем *домашнем каталоге*, но и в других *каталогах*. В этом случае будет удобно **сменить текущий каталог**. Для смены *текущего каталога* командной оболочки используется команда cd (от англ. "change directory" - "сменить *каталог*"). Команда cd принимает один параметр: имя *каталога*, в который нужно переместиться - сделать текущим. В качестве имени *каталога* можно использовать полный или *относительный путь*. В *приглашении командной строки* часто указывается *текущий каталог* shell - чтобы пользователю легче было ориентироваться, в каком *каталоге* он "находится" в данный момент.

Командная оболочка умеет **достраивать** имена файлов и *каталогов*: пользователю достаточно набрать несколько первых символов имени файла или *каталога* и нажать Tab. Если есть только один вариант завершения имени - оболочка закончит его сама, и пользователю не придется набирать оставшиеся символы. Достраивание - весьма существенное средство экономии усилий и повышения эффективности при работе с командной строкой. Современные командные оболочки умеют достраивать имена файлов и *каталогов*, а также имена команд. Достраивание наиболее развито в командном интерпретаторе zsh. Оболочка PowerShell также умеет достраивать имена.

Для перемещения в родительский каталог ("/home") удобно воспользоваться ссылкой "..". Необходимость вернуться в домашний каталог из произвольной точки файловой системы возникает довольно часто, поэтому командная оболочка поддерживает обозначение домашнего каталога при помощи символа "~". Поэтому чтобы перейти в домашний каталог из любого другого, достаточно выполнить команду "cd ~". При исполнении команды символ "~" будет заменен командной оболочкой на *полный путь* к *домашнему каталогу* пользователя.

При помощи символа "~" можно ссылаться и на *домашние каталоги* других пользователей: "~имя пользователя". Команда cd, поданная без параметров, эквивалентна команде "cd ~" и делает *текущим каталогом* домашний каталог пользователя.

1.7 Создание каталогов – утилита mkdir

В *домашнем каталоге*, как и в любом другом, можно создавать сколько угодно подкаталогов, в них - свои подкаталоги и т. д. Иными

словами, пользователю принадлежит фрагмент (поддерево) *файловой системы*, корнем которого является его *домашний каталог*.

Чтобы организовать такое поддерево, потребуется создать *каталоги* внутри домашнего. Для этого используется утилита mkdir. Она применяется с одним обязательным параметром: именем создаваемого *каталога*. По умолчанию *каталог* будет создан в *текущем каталоге*.

1.7.1 Создание нового пустого файла – команда touch

Для создания пустого файла с текущим временем создания служит команда touch имя_нового_файла. Для указания даты создания в формате ГГГГММДДhhmm используется ключ –t. Например
touch –t 0904080000 tst файл создан 8 апреля 2015 г.

1.8 Копирование и перемещение файлов

Для перемещения файлов и *каталогов* предназначена утилита mv (от англ. "move" - "перемещать"). У mv два обязательных параметра: первый - перемещаемый файл или *каталог*, второй - файл или *каталог* назначения. Имена файлов и *каталогов* могут быть заданы в любом допустимом виде: при помощи полного или относительного пути. Кроме того, mv позволяет перемещать не только один файл или *каталог*, а сразу несколько. За подробностями о допустимых параметрах и ключах следует обратиться к руководству по mv:

Перемещение файла внутри одной *файловой системы* в действительности равнозначно его **переименованию**: данные самого файла при этом остаются на тех же секторах диска, а изменяются *каталоги*, в которых произошло перемещение. Перемещение предполагает удаление ссылки на файл из того *каталога*, откуда он перемещен, и добавление ссылки на этот самый файл в тот *каталог*, куда он перемещен. В результате изменяется полное имя файла - *полный путь*, т. е. положение файла в *файловой системе*.

Иногда требуется создать копию файла: для большей сохранности данных, для того, чтобы создать модифицированную версию файла и т. п. В Linux для этого предназначена утилита cp (от англ. "copy" - "копировать"). Утилита cp требует использования двух обязательных параметров: первый - копируемый файл или *каталог*, второй - файл или *каталог* назначения. Как обычно, в именах файлов и *каталогов* можно использовать полные и *относительные пути*. Существует несколько вариантов комбинации файлов и *каталогов* в параметрах cp - о них можно прочесть в руководстве. Нужно иметь в виду, что в Linux утилита cp нередко настроена таким образом, что при попытке скопировать файл поверх уже существующего файла никакого предупреждения не выводится. В этом случае файл будет просто перезаписан, а данные,

которые содержались в старой версии файла, безвозвратно потеряны. Поэтому при использовании cp следует всегда быть внимательным и проверять имена файлов, которые нужно скопировать.

Созданная при помощи cp копия файла связана с оригиналом только в воспоминаниях пользователя, в файловой же системе исходный файл и его копия - две совершенно независимые и ничем не связанные единицы. Поэтому при наличии нескольких копий одного и того же файла в рамках **одной файловой системы** повышается вероятность запутаться в копиях или забыть о некоторых из них. Если задача состоит в том, чтобы обеспечить доступ к одному и тому же файлу из разных точек *файловой системы*, нужно использовать специально предназначенный для этого механизм *файловой системы* Linux - ссылки.

1.9 Файл и его имена: ссылки

1.9.1 Жесткие ссылки – утилита ln

Каждый файл представляет собой область данных на жестком диске компьютера или на другом носителе информации, которую можно найти **по имени**. В файловой системе Linux содержимое файла связывается с его именем при помощи *жестких ссылок*. Создание файла с помощью любой программы означает, что будет создана *жесткая ссылка* - имя файла, и открыта новая область данных на диске. Причем количество ссылок на одну и ту же область данных (файл) не ограничено, то есть у файла может быть несколько имен.

Пользователь Linux может добавить файлу еще одно имя (создать еще одну *жесткую ссылку* на файл) при помощи утилиты ln (от англ. "link" - "соединять, связывать"). Первый параметр - это имя файла, на который нужно создать ссылку, второй - имя новой ссылки. По умолчанию ссылка будет создана в *текущем каталоге*:

Пример 2. Создание жестких ссылок

```
[student@localhost ~]$ ln text text-hardlink
```

В **примере 2** в *домашнем каталоге* пользователя student создана жесткая ссылка с именем "text-hardlink" на файл "text". Если вывести подробный список файлов *текущего каталога* и его подкаталогов ("ls -lR"), то у файлов "text" и "text-hardlink" совпадут и размер, и время создания. Теперь "text-hardlink" и "text" - это два имени одного и того же файла.

Доступ к одному и тому же файлу при помощи нескольких имен может понадобиться в следующих случаях:

Одна и та же программа известна под несколькими именами.

Доступ пользователей к некоторым *каталогам* в системе может быть ограничен из соображений безопасности. Однако если все же нужно организовать доступ пользователей к файлу, который находится в таком

каталоге, можно создать *жесткую ссылку* на этот файл в другом *каталоге*.

Современные *файловые системы* даже на домашних персональных компьютерах могут насчитывать до нескольких десятков тысяч файлов и тысячи *каталогов*. Обычно у таких *файловых систем* сложная многоуровневая иерархическая организация - в результате пути ко многим файлам становятся очень длинными. Чтобы организовать более удобный доступ к файлу, который находится очень "глубоко" в иерархии *каталогов*, также можно использовать *жесткую ссылку* в более доступном *каталоге*. Полное имя некоторых программ может быть весьма длинным (например, i586-alt-linux-gcc-3.3), к таким программам удобнее обращаться при помощи сокращенного имени (*жесткой ссылки*) - gcc-3.3.

1.9.2 Индексные дескрипторы

Поскольку благодаря *жестким ссылкам* у файла может быть несколько имен, понятно, что вся существенная информация о файле в *файловой системе* привязана не к имени. В *файловых системах* Linux вся информация, необходимая для работы с файлом, хранится в *индексном дескрипторе*. Для **каждого** файла существует *индексный дескриптор*: не только для обычных файлов, но и для каталогов, *файлов-дырок* и т. д. Каждому файлу соответствует **один индексный дескриптор**.

Индексный дескриптор - это описание файла, в котором содержится:

- тип файла (обычный файл, *каталог*, специальный *файл* и т. д.);
- *права доступа* к файлу;
- информация о том, кому принадлежит файл;
- отметки о времени создания, модификации, последнего доступа к файлу;
- размер файла;
- указатели на физические блоки на диске, принадлежащие этому файлу - в этих блоках хранится "содержимое" файла.

Все *индексные дескрипторы* пронумерованы, поэтому номер *индексного дескриптора* - это уникальный идентификатор файла в *файловой системе* - в отличие от **имени** файла (*жесткой ссылки* на него), которых может быть несколько. Узнать номер *индексного дескриптора* любого файла можно при помощи утилиты ls с ключом –i

Если вывести номера *индексных дескрипторов* файла "text" и *жесткой ссылки* на него "text-hardlink" – можно увидеть, что эти номера совпадают, то есть этим двум именам соответствует один *индексный дескриптор*, т. е. один и тот же файл.

Все операции с *файловой системой* - создание, удаление и перемещение файлов - производятся на самом деле над *индексными*

дескрипторами, а имена нужны только для того, чтобы пользователь мог легко ориентироваться в файловой системе. Более того, имя (или имена) файла в его **индексном дескрипторе не указаны**. В *файловой системе* Ext2 имена файлов хранятся в *каталогах*: каждый *каталог* представляет собой список имен файлов и номеров их *индексных дескрипторов*. *Жесткую ссылку* (имя файла, хранящееся в *каталоге*) можно представлять как каталожную карточку, на которой указан номер *индексного дескриптора* - идентификатор файла.

Жесткая ссылка (hard link) - запись вида имя файла+номер *индексного дескриптора* в *каталоге*. *Жесткие ссылки* в Linux - основной способ обратиться к файлу по имени.

1.9.3 Символьные ссылки

У *жестких ссылок* есть два существенных ограничения:

- *Жесткая ссылка* может указывать только на файл, но не на *каталог*, потому что в противном случае в *файловой системе* могут возникнуть циклы - бесконечные пути.
- *Жесткая ссылка* не может указывать на файл в другой *файловой системе*. Например, невозможно создать на жестком диске *жесткую ссылку* на файл, расположенный на дискете. Чтобы избежать этих ограничений, были разработаны *символьные ссылки*. ***Символьная ссылка*** - это просто файл, в котором содержится имя другого файла. *Символьные ссылки*, как и жесткие, предоставляют возможность обращаться к одному и тому же файлу по разным именам. Кроме того, *символьные ссылки* могут указывать и на *каталог*, чего не позволяют *жесткие ссылки*. *Символьные ссылки* называются так потому, что содержат **символы** - *путь* к файлу или *каталогу*.

Символьная ссылка (symbolic link, файл-ссылка) - это файл особого типа ("l"), в котором содержится *путь* к другому файлу. Если на пути к файлу встречается *символьная ссылка*, система выполняет подстановку: исходный *путь* заменяется тем, что содержится в ссылке.
Символьную ссылку можно создать при помощи команды ln с ключом "-s" (сокращение от "symbolic").

Если выполнить команду cat имя_файла-ссылки, то на экран будет выведено содержимое файла, на который указывает ссылка.

Символьная ссылка вполне может содержать имя несуществующего файла. В этом случае ссылка будет существовать, но не будет "работать": например, если попробовать вывести содержимое такой "битой" ссылки при помощи команды cat, будет выдано сообщение об ошибке. Узнать, куда указывает *символьная ссылка*, можно при помощи утилиты realpath.

1.10 Удаление файлов и каталогов – утилиты rm и rmdir

В ОС Linux для удаления файлов предназначена утилита rm (сокращение от англ. "remove" - "удалять"):

Если удалить файл text в домашнем каталоге пользователя student, файл text-hardlink, который является *жесткой ссылкой* на удаленный файл text, сохранится, количество *жестких ссылок* на этот файл уменьшится с "2" до "1" - действительно, text-hardlink - теперь единственное имя этого файла. Однако если удалить и *жесткую ссылку* text-hardlink, у этого файла больше не останется ни одного имени, он станет недоступным пользователю и будет уничтожен.

Утилита rm предназначена именно для удаления *жестких ссылок*, а не самих файлов. В Linux, чтобы полностью удалить файл, требуется последовательно удалить все *жесткие ссылки* на него. При этом все *жесткие ссылки* на файл (его имена) равноправны - среди них нет "главной", с исчезновением которой исчезнет файл. Пока есть хоть одна ссылка, файл продолжает существовать. Впрочем, у большинства файлов в Linux есть только одно имя (одна *жесткая ссылка* на файл), поэтому команда rm имя файла в большинстве случаев успешно удаляет файл.

Как уже говорилось, *символьные ссылки* - это отдельные файлы, поэтому после удаления файла text, text-symlink, который ссылался на этот файл, продолжает существовать, однако теперь это - "битая ссылка", поэтому его также можно удалить командой rm.

Для удаления *каталогов* предназначена другая утилита - rmdir (от англ. "remove directory"). Впрочем, rmdir согласится удалить *каталог* только в том случае, если он пуст - в нем нет никаких файлов и подкаталогов. Удалить *каталог* вместе со всем его содержимым можно командой rm с ключом "-r" (recursive). Команда rm -r *каталог* - очень удобный способ потерять в одночасье **все** файлы: она рекурсивно обходит весь *каталог*, удаляя все, что попадется: файлы, подкаталоги, *символьные ссылки*... а ключ "-f" (force) делает ее работу еще неотвратимее, так как подавляет запросы вида "удалить защищенный от записи файл", так что rm работает безмолвно и безостановочно.

ПОМНИТЕ: если вы удалили файл, значит, он уже не нужен, и не подлежит восстановлению!

В Linux не предусмотрено процедуры восстановления удаленных файлов и *каталогов*. Поэтому стоит быть **очень** внимательным, отдавая команду rm и, тем более, rm -r: нет никакой гарантии, что случайно удаленные данные удастся восстановить.

1.11 Права доступа в файловой системе

1.11.1 Идентификатор пользователя

Говоря о *правах доступа* пользователя к файлам, заметим, что в действительности манипулирует файлами не сам пользователь, а запущенный им *процесс* (например, утилита rm или cat). Поскольку и файл, и *процесс* создаются и управляются системой, ей нетрудно организовать какую угодно политику доступа одних к другим, основываясь на любых свойствах *процессов* как субъектов и файлов как объектов системы.

В Linux, однако, используются не какие угодно свойства, а результат *идентификации* пользователя – его UID. Каждый *процесс* системы обязательно **принадлежит** какому-нибудь пользователю, и *идентификатор пользователя* (UID) – обязательное свойство любого *процесса* Linux. Когда программа login запускает стартовый командный интерпретатор, она приписывает ему UID, полученный в результате диалога. Обычный запуск программы (exec()) или порождение нового *процесса* (fork()) не изменяют UID процесса, поэтому **все** процессы, запущенные пользователем во время терминальной сессии, будут иметь его идентификатор.

Поскольку UID однозначно определяется входным именем, оно нередко используется **вместо** *идентификатора* – для наглядности. Например, вместо выражения "*идентификатор пользователя*, соответствующий *входному имени* student", говорят "UID student" (в приведенном ниже примере этот *идентификатор* равен **500**):

Пример 3. Как узнать идентификаторы пользователя и членство в группах

```
[student@localhost student]$ id
 uid=500 (student) gid=500(student) группы=500 (student)
```

Утилита id выводит *входное имя* пользователя и соответствующий ему UID, а также *группу по умолчанию* и полный список *групп*, членом которых он является.

1.11.2 Идентификатор группы

Пользователь может быть членом нескольких *групп*, равно как и несколько пользователей могут быть членами одной и той же *группы*. Исторически сложилось так, что одна из *групп – группа по умолчанию* – является для пользователя основной - когда говорят о "GID пользователя", имеют в виду именно идентификатор *группы по умолчанию*. GID пользователя вписан в *учетную запись* и хранится в /etc/passwd, а информация о соответствии **имен** *групп* их идентификаторам, равно как и о том, в какие **еще** *группы* входит пользователь – в файле /etc/group. Из этого следует, что пользователь **не может не быть** членом как минимум

одной *группы*.

1.11.3 Ярлыки объектов файловой системы

При создании объектов файловой системы – файлов, каталогов и т. п. – каждому приписывается *ярлык*. ***Ярлык*** включает в себя UID – *идентификатор пользователя-хозяина файла*, GID – идентификатор *группы*, которой принадлежит файл, тип объекта и набор так называемых *атрибутов* (код доступа), а также некоторую дополнительную информацию. *Атрибуты* (или код доступа) определяют, кто и что имеет право делать с файлом, они описаны ниже:

Пример 4. Атрибуты каталогов, показанные командой ls -l

итого 88

drwxr-xr-x	2 root root	4096 Апр	4 2015 bin
drwxr-xr-x	4 root root	4096 Апр	4 2016 boot
drwxr-xr-x	10 root root	3520 Апр	5 14:26 dev
drwxr-xr-x	90 root root	8192 Апр	5 14:22 etc
drwxr-xr-x	3 root root	4096 Апр	4 21:22 home
drwxr-xr-x	11 root root	4096 Апр	4 2016 lib
drwx------	2 root root	16384 Апр	4 2016 lost+found
drwxr-xr-x	4 root root	4096 Апр	5 14:22 media
drwxr-xr-x	2 root root	4096 Июл	11 2015 misc
drwxr-xr-x	2 root root	4096 Окт	20 2016 mnt
drwxr-xr-x	2 root root	0 Апр	5 14:21 net
drwxr-xr-x	2 root root	4096 Окт	20 2016 opt
dr-xr-xr-x	106 root root	0 Апр	5 2015 proc
drwxr-x---	31 root root	4096 Апр	5 14:29 root
drwxr-xr-x	2 root root	8192 Апр	4 2016 sbin
drwxr-xr-x	2 root root	4096 Окт	20 2016 selinux
drwxr-xr-x	2 root root	4096 Окт	20 2015 srv
drwxr-xr-x	11 root root	0 Апр	5 2016 sys
drwxrwxrwt	16 root root	4096 Апр	5 14:26 tmp
drwxr-xr-x	15 root root	4096 Апр	4 2015 usr
drwxr-xr-x	21 root root	4096 Апр	4 2015 var

Ключ "-l" утилиты ls определяет длинный (**l**ong) формат выдачи (справа налево): имя файла, время последнего изменения файла, размер в байтах, группа, хозяин, количество *жестких ссылок* и строчка *атрибутов*. Первый символ в строчке *атрибутов* определяет тип файла. Тип "-" отвечает "обычному" файлу, а тип "d" – каталогу (**d**irectory).

Несмотря на то, что создание *жестких ссылок* на каталог невозможно, значение поля "количество *жестких ссылок*" (второй столбец) для всех каталогов примера равно **двум**, а не одному. На самом деле этого и следовало ожидать, потому что **любой** каталог файловой

системы Linux всегда имеет не менее двух имен: собственное (например, tmp) и имя "." в самом этом каталоге (tmp/.). Если же в каталоге создать подкаталог, количество *жестких ссылок* на этот каталог увеличится на 1 за счет имени ".." в подкаталоге (например, tmp/subdir1/..):

1.11.4 Иерархия прав доступа

Рассмотрим более подробно, чему соответствуют девять символов в строке *атрибутов*, выдаваемой ls. Эти девять символов имеют вид "rwxrwxrwx", где некоторые "r", "w" и "x" могут заменяться на "-". Очевидно, буквы отражают принятые в Linux три вида доступа – чтение, запись и использование – однако в *ярлыке* они присутствуют в трех экземплярах!

Дело в том, что любой пользователь (*процесс*) Linux по отношению к любому файлу может выступать в трех **ролях**: как *хозяин* (**u**ser), как член *группы*, которой принадлежит файл (**g**roup), и как *посторонний* (**o**ther), никаких отношений собственности на этот файл не имеющий. Строка *атрибутов* – это три тройки "rwx", описывающие *права доступа* к файлу *хозяина* этого файла (первая тройка, "u"), *группы*, которой принадлежит файл (вторая тройка, "g") и *посторонних* (третья тройка, "o"). Если в какой-либо тройке не хватает буквы, а вместо нее стоит "-", значит, пользователю в соответствующей роли будет в соответствующем виде доступа отказано.

При выяснении отношений между файлом и пользователем, запустившим *процесс*, роль определяется так:
Если *UID* файла совпадает с *UID процесса*, пользователь – *хозяин файла*
Если *GID* файла совпадает с *GID* **любой** *группы*, в которую входит пользователь, он – член *группы*, которой принадлежит файл.
Если ни *UID*, ни *GID* файла не пересекаются с *UID процесса* и списком *групп*, в которые входит запустивший его пользователь, этот пользователь – *посторонний*.
Именно в роли *хозяина* пользователь (*процесс*) может **изменять** *ярлык файла*. Единственное, чего не может делать *хозяин* со своим файлом – менять ему *хозяина*.

1.12 Использование прав доступа в Linux

1.12.1 Использование групп

В Linux определено несколько *системных групп*, задача которых – обеспечивать доступ членов этих *групп* к разнообразным ресурсам системы. Часто такие *группы* носят говорящие названия: "disk", "audio", "cdwriter" и т. п. Если обычным пользователям доступ к некоторому файлу, каталогу или специальному файлу Linux закрыт, он открыт членам *группы*, которой этот объект принадлежит.

Например, в Linux почти всегда используется *виртуальная файловая система* /proc – каталог, в котором в виде подкаталогов и файлов представлена информация из *таблицы процессов*. Имя подкаталога /proc совпадает с *PID* соответствующего *процесса*, а содержимое этого подкаталога отражает свойства *процесса*. *Хозяином* такого подкаталога будет хозяин *процесса* (с правами на чтение и использование), поэтому **любой** пользователь сможет посмотреть информацию о **своих** *процессах*. Именно каталогом /proc пользуется утилита ps. Использование утилиты ps для просмотра выполняющихся процессов Linux рассматривается в работе “Процессы ОС Linux“.

1.13 Суперпользователь

Суперпользователь - единственный пользователь в Linux, на которого не распространяются ограничения *прав доступа*. Имеет нулевой *идентификатор пользователя*.

Суперпользователь в Linux – это **выделенный** пользователь системы, на которого **не распространяются** ограничения *прав доступа*. *UID* суперпользовательских *процессов* равен 0: так система отличает их от *процессов* других пользователей. Именно суперпользователь имеет возможность произвольно изменять владельца и *группу* файла. Ему открыт доступ на чтение и запись к **любому файлу** системы и доступ на чтение, запись и использование к **любому каталогу**. Наконец, суперпользовательский *процесс* может на время сменить **свой собственный** *UID* с нулевого на любой другой. Именно так и поступает программа login, когда, проведя процедуру идентификации пользователя, запускает стартовый командный интерпретатор.

Среди *учетных записей* Linux всегда есть запись по имени root ("корень"), соответствующая нулевому идентификатору, поэтому вместо "суперпользователь" часто говорят "root". Множество системных файлов принадлежат root, множество файлов только ему доступны на чтение или запись. Пароль этой учетной записи – одна из самых больших драгоценностей системы. Именно с ее помощью системные администраторы выполняют самую ответственную работу.

Существует два различных способа получить права суперпользователя. Первый – это зарегистрироваться в системе под этим именем, ввести пароль и получить *стартовую оболочку*, имеющую нулевой *UID*. Это – самый неправильный способ, пользоваться которым стоит, только если нельзя применить другие. В ОС Ubuntu описанный способ не используется, вместо него используется второй способ.

Второй способ —воспользоваться специальной утилитой sudo, которая позволяет выполнить одну или несколько команд от лица другого пользователя. По умолчанию эта утилита выполняет команду от лица

пользователя root, то есть запускает командный интерпретатор с нулевым *UID*. Отличие от предыдущего способа в том, что всегда известно, кто именно запускал sudo, а значит, ясно, с кого спрашивать за последствия.

1.14 Поиск файлов

Для поиска файла по имени или его части используется утилита locate. Параметр задает имя файла. Для поиска без учета регистра служит ключ –i.

Для ограничения объема выводимой информации используется ключ –n число. Построчный вывод получается, если результаты поиска направить по конвейеру в программу less, например

locate mp3 | less

Утилита locate ведет поиск в базе данных, которая должна периодически обновляться утилитой updatedb, выполняемой с правами администратора.

Другой способ найти файл предоставляет утилита **find.** Ее ключи приведены в (табл. 1).

Таблица 1.

Ключи утилиты find

Ключ	Назначение
-name	Задает имя файла или его часть
-size	Задает размер файла, например 12k
-type	Задает тип объекта для поиска: f-обычный файл d-каталог l-символьная ссылка
-a	Логическая связка and
-o	Логическая связка or
-user	Задает имя пользователя

Достоинствами утилиты find являются независимость от базы данных и широкие функциональные возможности, недостаток – меньшая скорость поиска по сравнению с locate.

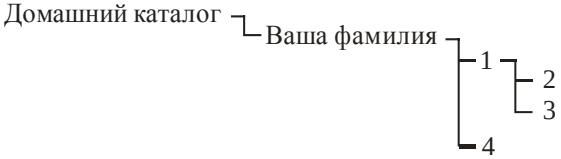
2 МЕТОДИКА ВЫПОЛНЕНИЯ

1. Ознакомиться с теоретическими сведениями.
2. После загрузки ОС Linux и запроса имени ввести имя и пароль пользователя.
3. По окончании загрузки ОС запустить терминал.

Все задания работы **следует** выполнить в режиме командной строки с помощью терминала.

Задания:

1. Создать в домашнем каталоге следующую структуру подкаталогов (существующие каталоги не удалять!):



2. Скопировать файл **/etc/group** в каталоги **1**, 2, 3 и 4 используя абсолютные имена копируемого файла и каталога назначения.
3. С помощью утилиты file вывести на экран сведения о 3 - 4 различных файлах (в том числе из каталогов /bin и /dev).
4. Выполнить команду ls –l /dev используя таблицу 2 обозначений типов файлов

Таблица 2.

Обозначения типов файлов

Символ	Тип файла
d	Каталог
l	Символьная ссылка
s	Сокет
b	Блочное устройство
c	Символьное устройство
p	Именованный канал

перечислить типы файлов, хранящихся в каталоге /dev

5. Используя справочную систему, ознакомиться с ключами утилиты ls -R, -1 (единица), -m, --color, ключи, определяющие порядок вывода на экран
6. Создать жесткую и символическую ссылки для одного из созданных в п.2 файлов.

Таблица 3.

Индивидуальные задания для бригад

Номер бригады	Задание
1	Вывести список имен файлов из /var, используя ключ –l Список упорядочить по размерам файлов. 2. Найти файлы, имена которых оканчиваются на pdf
2	Вывести список имен файлов из /bin, используя ключ –l Список упорядочить по датам создания

Номер бригады	Задание
	2. Найти файлы, имена которых оканчиваются на jpg
3	Вывести список имен файлов из /sbin, используя ключ –l Список упорядочить по именам 2. Найти файлы, размеры которых превышают 25к (запись +25k)
4	Вывести список имен файлов из /tmp, используя ключ –l Список упорядочить по именам 2. Найти файлы, имена которых оканчиваются на text
5	Вывести список имен файлов из /usr, используя ключ –l Список упорядочить по размерам файлов. 2. Найти файлы, имена которых оканчиваются на jpg и размеры более 1к
6	Вывести список имен файлов из /bin, используя ключ –l Список упорядочить по датам создания 2. Найти файлы, размеры которых превышают 15к (запись +15k)
7	Вывести список имен файлов из /usr, используя ключ –l Список упорядочить по размерам файлов. 2. Найти файлы, размеры которых превышают 25к (запись +25k) и имена начинаются на s
8	Вывести список имен файлов из /var, используя ключ –l Список упорядочить по датам создания 2. Найти файлы, размеры которых превышают 25к (запись +25k) и имена начинаются на s, а заканчиваются на jpg
9	Вывести список имен файлов из /sbin, используя ключ –l Список упорядочить по размерам файлов 2. Найти файлы, размеры которых превышают 1М (запись +1m)
10	Вывести список имен файлов из /bin, используя ключ –l Список упорядочить по именам 2. Найти файлы, размеры которых превышают 5к (запись +5k)

7. Выключить компьютер.

3 ОТЧЕТ О РАБОТЕ

Готовится в письменном виде один на бригаду. Содержание отчета:

- построенное в задании 1 дерево каталогов.
- описания назначений ключей команды ls.

- результаты выполнения заданий.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Типы файлов ОС Linux
2. Назначение утилиты file.
3. Структура дерева каталогов ОС Linux.
4. Отличия структуры файловых систем ОС Windows и Linux.
5. В чем отличие каталогов /var и /tmp.
6. Назначение утилиты pwd.
7. Назначение утилиты cat.
8. Назначение утилиты ls. Использование ключей –F, -a.
9. Утилита mkdir.
10. Утилиты копирования и перемещения файлов.
11. Жесткие ссылки: назначение и создание.
12. Создание файлов.
13. Символьные ссылки.
14. Удаление файлов и каталогов. Как восстановить ошибочно удаленный файл?
15. Назначение утилиты id.
16. Ярлыки объектов файловой системы.
17. Права доступа к файлу.
18. Суперпользователь и его права.
19. Назначение утилиты sudo.
20. Утилиты поиска файлов locate и find, их достоинства и недостатки.