*ПЗ-03. Дополнительно:* ***Работа с файловыми системами*** (больше теории)

***Файл*** (англ. file) — *именованная область данных на носителе информации*.

Работа с файлами реализуется средствами операционных систем. Многие ОС приравнивают к файлам и обрабатывают сходным образом и другие ресурсы:

* области данных (необязательно на диске);
* устройства — как физические, например, порты или принтеры, так и виртуальные (/dev/null, /dev/random, /dev/urandom);
* потоки данных (именованный канал);
* сетевые ресурсы, сокеты;
* прочие объекты операционной системы.

**Файловая система (ФС). Основные аспекты:**

1. Во-первых, ФС – это набор правил и конструкций, описывающих то, как сохраняются файлы на диске.

2. Во-вторых, ФС – это совокупность всех файлов, хранимых в компьютере.

3. В-третьих (и это значение термина характерно именно для UNIX-систем) ФС – это совокупность всех файлов на разделе диска или устройстве + сами устройства

Т.о., **ФС** – *это порядок, определяющий способ организации, хранения и именования информации на устройствах хранения, а только потом практическая реализация этого порядка*.

Т.е. первичен некий свод правил: что где лежит, как называется и т.д. и т.п., а практические реализации файловых систем, например, NTFS или ext4 – это технический способ организации информации на определенном типе носителя в соответствии с принятыми правилами.

**Отличия файловой системы Linux от Windows.**

В Linux нет дисков C или D. Один физический диск (или несколько) при установке системы разбивается на каталоги и подкаталоги. Основной, корневой каталог обозначается символом / (слэш) Вместо файла подкачки существует отдельный раздел /swap.

**Типы файлов LINUX** NB: См. результаты команды ls –l (первый символ):

* Обычный файл (–)
* Каталог (d)
* Файл – устройство:
  + Символьное (c)
  + Блочное (b)
* Именованный канал FIFO (p)
* Сокет (s)
* Ссылка (l)

**Типы файловых систем в Linux:**

* Локальная файловая система:
  + Псевдо - файловые системы (proc, sysfs)
  + «Нормальные» (disk-based) файловые системы (ext2, ext3)
* Распределённая (сетевая) файловая система (NFS (Sun Microsystems))

Структура файловой системы

|  |  |
| --- | --- |
| **Windows** | **Linux** |
| linux-filesystem-001.jpg | linux-filesystem-002.jpg |

В Linux – все есть файл, дисковые накопители – это тоже определенный тип файла.

Кроме того, ФС Linuх иерархична, т.е. имеет один единственный корень, обозначаемый символом /.

В Linux (см. рис. справа) также имеется системный диск, на котором расположен корень ФС и все системные папки. Пользовательские данные все также располагаются на втором физическом диске, но в системе вы его не увидите. Почему? Да потому что в ФС Linux такого понятия нет. Новички на этом месте обычно впадают в панику, считают, что диск не определился и начинают судорожно искать его где можно и где нельзя.

Но **диск, с точки зрения Linux, это тоже файл** и может быть расположен в любом месте файловой системы. В нашем случае никто не мешает поместить его на место папки /home, где располагаются пользовательские данные. Такое действие называется монтированием, а место файловой системы, к которому присоединен носитель - точкой монтирования. Т.о., папка /home целиком окажется на втором жестком диске, что будет в какой-то мере аналогично переносу пользовательских данных на диск D.

Съемные носители, такие как компакт-диски, флешки и т.п. в графической среде монтируются автоматически в предопределенную директорию /media, а в рабочем окружении появляется ярлык, что делает работу с ними неотличимой от Windows. В серверной среде вам потребуется монтировать съемные носители вручную, также вы можете выбрать произвольную точку монтирования, но лучше не изобретать велосипед, а использовать /media.

Еще один момент, связанный с **иерархичностью ФС Linuх** – это **неизменность путей к данным при изменении физической структуры дисков**.

Простой пример: у вас есть большая коллекция музыки, которую вы решили вынести на отдельный жесткий диск. В Windows все просто: купили новый диск, подключили, скопировали. Но есть один недостаток. Если до этого пути к музыке были D:\Музыка, то стали E:\Музыка и все ранее созданные ярлыки, плейлисты и т.п. стали неверны.

В Linux процесс выглядит немного посложнее:

* новый диск временно монтируется, скажем, в /mnt,
* затем на него переносится содержимое папки /home/Музыка,
* после чего он монтируется на постоянной основе в точку /home/Музыка.

В итоге наша коллекция лежит на отдельном жестком диске, но все плейлисты как работали с /home/Музыка так и продолжают работать.

Эту возможность трудно переоценить, особенно когда надо вынести на отдельный раздел не коллекцию музыки, а базу почтового сервера или содержимое хранилища виртуальных машин.

**Имена файлов и папок в Linux** ограничены длинной в 256 символов и запретом на / (слеш). Отдельно следует упомянуть о символах **.** (точка) и **~** (тильда), точка перед именем файла или директории добавляют ему атрибут "**скрытый**", а тильда после имени добавляет атрибут "**архивный**" и также делает файл скрытым. Например, **.file** – *скрытый файл*, а **file~** – *архивный*.

Но это не говорит о том, что можно называть файлы, как вам нравится, без оглядки на иные системы, прежде всего Windows. Допустим, мы решили вопреки всем правилам назвать файл **<Правила?>**, в Linux нам никто не помешает этого сделать, но уже попытавшись скопировать его на флешку с FAT32 нас постигнет наказание, имя файла превратится в набор подчеркиваний.

|  |  |
| --- | --- |
| Поэтому, если вы не хотите превращения имени вашего файла в нечто совершенно неожиданное, то не используйте в именах символы:  **\ / : \* ? " < > |**  Также не злоупотребляйте служебными символами в начале имени, например, никто не мешает создать вам файл с именем **-text**, однако при попытке скопировать его в консоли вы получите неожиданный результат: | linux-filesystem-005.jpg |



Еще одна важная особенность – **Linux – регистрозависимая система**, это означает, что *file.txt, File.txt и file.TXT – это три разных файла!!!*

Отдельный разговор – это **пробелы в именах файлов**. Такая практика не запрещается, но считается в профессиональных кругах дурным тоном. По возможности избегайте таких имен.

**Жесткие и символические ссылки**

Начав работать с Linux, вы обязательно столкнетесь с этим типом файлов. Они не имеют прямого аналога в файловой системе Windows и поэтому на них стоит остановиться подробнее.

Начнем с **символических ссылок**, в первом и достаточно грубом приближении они напоминают *ярлыки Windows*, **это специальный тип файла, который служит указателем на другой файл**. Но при этом, в отличии от ярлыка, воспринимается системой прозрачно, т.е. не как файл ярлыка, а как файл типа, на который указывает ссылка. Проще говоря, мы не можем подсунуть Windows приложению ярлык на библиотеку вместо библиотеки, а в Linux это общепринятая практика.

Разберем устройство символических ссылок подробнее, в дальнейшем это поможет понять их отличие от жестких ссылок и лучше понять сферу их применения. Как известно **файл** (или **каталог**) есть ничто иное, как *некая именованная область на диске*. Соответствие имени файла областям диска хранится в специальной таблице файловой системы (inode в Linux, MFT или FAT в Windows), т.е. ***файл*** *– это указатель в inode на определенную область данных на диске.*

|  |  |
| --- | --- |
| **Символическая ссылка** – *это отдельный файл, со своим inode, который указывает на изначальный файл.*  Если первоначальный файл будет удален или перемещен, символические ссылки останутся, но окажутся бесполезными.  В тоже время мы можем как угодно перемещать символические ссылки или удалять их, что никак не скажется на первоначальном файле.  **Символические ссылки** широко применяются в случаях, *когда один и тот-же файл должен быть доступен под разными именами или в разных местах, но при этом его содержимое не должно меняться*. | 1233.png |

Например, некое приложение требует библиотеку lib-1.0.1.so, в то время как есть совместимая библиотека lib-1.0.5.so, в этом случае создаем символическую ссылку на lib-1.0.5.so с именем lib-1.0.1.so и проблема решена. Другой случай, приложение требует lib-1.0.5.so, которая есть в системе, но ищет ее в другом месте, в этом случае делаем символическую ссылку с таким же именем, но в нужное расположение.

Визуально символические ссылки можно определить по значку ярлычка в графическом окружении или по символу @ перед именем файла в mc (но, если вы создадите файл с началом имени на @ - символической ссылкой он не станет).

Другое применение – это *файлы настроек*. Например, веб-сервер Аpache содержит два набора директорий: с доступными модулями и настройками, и применяющимися в текущий момент. Для того, чтобы подключить модуль или настройку достаточно сделать символическую ссылку из одной директории в другую, надо выключить - удаляем символическую ссылку. Сам файл модуля или настроек остается на своем месте и может быть использован в дальнейшем.

Символические ссылки можно создавать не только на файлы, но и на каталоги. Также нет ограничения на физическое расположение символических ссылок в пределах одной физической файловой системы (одного раздела).

**Жесткие ссылки**. Это принципиально иная сущность.

|  |  |
| --- | --- |
| **Жесткая ссылка** *создает еще одну запись в inode файла, т.е. по сути является еще одним его именем, а внешне является полной копией первоначального файла, определить при этом где файл, а где жесткая ссылка практически невозможно*.  Отличие **жесткой ссылки** от **копии** заключается в том, что несмотря на разные имена, это один и тот-же файл и, если изменить одно из его имен - изменятся остальные.  Также файл физически существует до тех пор, пока на него есть хоть одна ссылка (первоначальный это файл или символическая ссылка – значения не имеет). | linux-filesystem-014.jpg |

*Для чего могут понадобиться жесткие ссылки?*

* Самое распространенное их применение, это создание полных копий файла без лишних затрат дискового пространства.
* Например, инсталляционные пакеты.
* Допустим нам надо выложить на FTP несколько вариантов ПО, в каждый из наборов которого входят некие общие пакеты и документация, причем нужна синхронизация между этими файлами, например, если мы внесли изменения в документацию, это должно отразиться во всех наборах.
* Жесткие ссылки отлично решают эту задачу.

В тоже время **жесткие ссылки** создают дополнительную путаницу, особенно если вам нужно удалить файл во всех местах использования. Поэтому подходите к использованию жестких ссылок с осторожностью и не злоупотребляйте ими, так как в большинстве случаев символических ссылок более чем достаточно.

**Жесткие ссылки**, так как являются дополнительной записью в inode, могут использоваться только в пределах одного физического раздела. Также спецификации POSIX запрещают создание жестких ссылок для каталогов.

**Права доступа**

Еще одним неотъемлемым атрибутом любой файловой системы являются права доступа к файлам и папкам. Linux унаследовал классическую UNIX-систему прав, они не так гибки, как хотелось бы, но обеспечивают приемлемый уровень гибкости и безопасности для простых систем.

Права доступа к файлу (а как мы помним, в Linux все есть файл) хранятся в специальном 16-битовом поле атрибутов файла:

|  |  |
| --- | --- |
|  | Первые четыре бита устанавливают флаг типа объекта, они задаются при создании файла и не могут быть изменены. Флаг может иметь следующие значения: |

- – Отсутствие флага - обычный файл

l – Символическая ссылка

d – Директория

b – Блочное устройство

c – Символьное устройство

p – Канал, устройство fifo

s – Unix-сокет

Следующие три бита хранят особые признаки, влияющие на запуск исполняемых файлов и некоторые иные права, к ним мы вернемся несколько позже. И наконец следующие девять бит, разделенные на блоки по три бита содержат права доступа к файлу или директории.

Каждый файл в UNIX должен иметь владельца (пользователь, user), группового владельца (группа, group) и остальных пользователей (остальные, other), каждый из этих пользователей может иметь права на чтение (r), запись (w) и исполнение (x).

Применительно к каталогам флаги имеют несколько иной смысл:

r – право получения имен файлов в каталоге (но не их атрибутов),

x – право получения доступа к файлам и их атрибутам,

w – право манипулировать именами файлов, т.е. создавать, удалять, переименовывать.

Минимальный разумный набор прав на каталог: rx-, так как только r позволит получить только имена файлов, но не их тип, размер и т.п. Ниже показан пример папки с набором прав r--.

Как нетрудно заметить, w без х не имеет никакого смысла и равносильно его отсутствию.

|  |  |
| --- | --- |
| Запись прав может производиться как в символьной, так и в числовой форме, для этого используют двоичное или восьмеричное (что удобнее) значение установленных битов.  В то время, как в системе используются преимущественно символьные обозначения, для целей администрирования обычно используются |  |

|  |  |
| --- | --- |
| цифровые восьмеричные значения. Потому что проще, быстрее и удобнее написать, что права на файл должны быть **644**, а не **rw-r--r--.**  В тоже время символьная запись позволяет быстро оценить реальный набор прав, а не вспоминать, что значит 755. Стандартные права для вновь создаваемых файлов - 664, папок - 775, исключение - файлы и каталоги, созданные с правами суперпользователя, они получают 644 и 755 соответственно.  На практике из всех сочетаний флагов доступа реально используются только 0, 4, 5, 6, 7 для файлов и 0, 5, 7 для папок.  Разобравшись с основными правами перейдем к особым признакам, таких три:    Начнем с младшего бита, его установка означает установку **sticky-бита** для каталога, установка данного флага для файлов в современных системах игнорируется. Дословно sticky обозначает "липкий", что довольно хорошо соответствует его смыслу. |  |

После установки данного флага удалить файл из каталога может только его владелец или суперпользователь, даже если на файлы и папку стоят права **777**.

|  |  |
| --- | --- |
| Эта опция может быть использована для организации файловых серверов типа "*файлопомойка*", когда нужно организовать доступ для всех, без разбора, но исключить возможность случайного или преднамеренного удаления чужих файлов. Установить sticky-бит может только суперпользователь, снять - суперпользователь или владелец каталога. |  |

Опции SUID и SGID позволяют любому пользователю запускать файл на исполнение с правами его владельца или группы. Для чего это нужно? В обычных условиях файл запускается с правами текущего пользователя, что не всегда достаточно для его работы. Например, таким образом работает утилита passwd, нам нужно чтобы пользователь имел возможность изменить свой пароль без повышения прав, но данная операция требует прав суперпользователя. Как быть? Использовать признак SUID. Если мы проверим права на утилиту, то увидим запись rwsr-xr-x или 4755.

При установке признаков SUID и SGID они заменяют символ x на s в соответствующей группе символьного представления или записываются перед основными правами в восьмеричном виде. Вообще-то в цифровом виде все права следует записывать в четырехзначном формате, так как если особые признаки не установлены, то это 000 BIN или 0 OCT. Т.е. правильно писать не 777, а 0777, но обычно для краткости первый ноль опускают.

Если установлены несколько признаков, то записывается восьмеричное число аналогичное установленным битам в двоичном формате, например, SUID + sticky это 101 BIN или 5 OCT. Установленный sticky-бит заменяет x на t в группе other. Ниже показан каталог с правами 7775 или rwsrwsr-t, что соответствует установке на него SUID, SGID и sticky-бит одновременно.

|  |  |
| --- | --- |
| Данная запись сделана нами исключительно в тестовых целях, так как установка SUID для каталога не имеет смысла, а установка SGID приведет к тому, что групповым владельцем создаваемых в нем файлов будет группа владельца каталога, а не группа создавшего его пользователя, как происходит по умолчанию. |  |

Также, ввиду потенциальной опасности, игнорируется установка SUID и SGID для скриптов.

В графической среде установка прав разнится в зависимости от выбранного настольного окружения, например, в Unity настройки выполнены в понятной пользователю форме, но особые признаки не отображаются и не могут быть установлены, а для папок присутствует бесполезный набор прав r--.

И наконец KDE сочетает достаточно лаконичный основной набор прав с возможностью установки признака "исполняемый" для файлов и sticky-бит для папок с возможностью явно указывать особые признаки в дополнительных настройках.

В любом случае следует помнить, что графический интерфейс не является основным средством администрирования Linux, а представляет надстройку над средствами командной строки.

|  |  |
| --- | --- |
| linux-filesystem-020.jpg | linux-filesystem-021.jpg |