СПО

Лабораторная работа №15

«Файловая система ОС Linux»

Цель: Изучить команды работы с файловой системой ОС Linux.

Работая в любой графической оболочке (например, KDE или Gnome) мы часто выполняем такие операции как копирование, перемещение, переименование объектов (файлов и папок), также создаем файлы и каталоги. Все это изменяет файловую структуру (по крайней мере, ее "пользовательскую" часть). Очевидно, что подобные операции (копирование, перемещение и др.) должны быть предусмотрены и в командной оболочке Linux.

копирование

Для копирования файлов в bash используется команда **cp** (от "copy"– копировать), которая имеет два обязательных аргумента: имя (адрес) исходного файла и имя (адрес) создаваемой копии или адрес каталога, куда помещается копия.

cp *адрес/имя_оригинала* *адрес/имя _копии*

или

cp *адрес/имя_оригинала* *адрес/*

Адрес может быть как абсолютным, так и относительным. Если операции с файлами выполняются в текущем каталоге, то адрес нет смысла указывать, а пишется только имя исходного файла и имя копии. При этом следует помнить, что файлов с одинаковыми именами и адресами не может быть, поэтому имя копии должно отличаться от имени исходного файла.

Рассмотрим несколько примеров.

cp readme readme2

В данном случае создается копия файла readme, которая остается в той же директории под именем readme2.

cp readme Desktop/

Создается копия на рабочем столе. Непосредственное имя копии в данном случае можно оставить прежним, т.к. полные имена (адрес + имя) файлов различны. Используется

относительная адресация (каталог Desktop является дочерним по отношению к домашнему каталогу).

cp /home/irina/tux.png /mnt/D/pingvin.png

Здесь копируется файл из домашнего каталога пользователя irina в каталог D, используются абсолютные адреса, имя копии изменяется.

Чтобы скопировать каталог необходимо после команды cp прописать ключ r:

cp -r ./letters ./oldletters

перемещение и переименование

Для перемещения и переименования в bash-оболочки Linux используется одна команда – **mv** (от "move" – перемещать). Также как и с командой копирования обязательно наличие двух аргументов, а выбор того или иного действия (перемещения или переименования) зависит от того, что это за аргументы. При перемещении файл меняет свой адрес, следовательно, если в первом и втором аргументах указаны различные адреса, то произойдет перемещение, если одинаковые (а различны только имена) – переименование.

В случае перемещения во втором аргументе может быть указан только каталог (перемещение без переименования), куда требуется переместить файл.

mv document.txt Work

mv document.txt Work/doc23.txt

Здесь в первом случае файл document.txt перемещается в каталог Work, а во-втором случае одновременно происходит перемещение файла и его переименование: файл document.txt перемещается в директорию Work и получает новое имя doc23.txt.

При использовании команды mv для переименования в качестве второго аргумента указывается новое имя:

mv order.txt orderNew.txt

mv Work/list.odt Work/names.odt

создание файлов и каталогов

Новые директории создаются командой mkdir. Например, чтобы создать каталог Work в текущей директории необходимо выполнить следующую команду:

mkdir Work

или

mkdir ./Work

Существует множество способов создания файлов. Один из них – это создание пустого файла с помощью команды touch. В качестве аргумента ей передается имя файла. Еще один способ – это перенаправление вывода какой-либо команды в файл. Пример:

cal > ./Work/January

Здесь команда cal выводит календарь на текущий месяц, а поскольку стоит знак ">" (в данном случае обозначающий "направить результат выполнения команды в ..."), то вывод команды будет записан в файл January, находящийся в папке Work.

удаление файлов и каталогов

Для удаления каталогов используется команда rmdir. Удалять можно только пустые каталоги, т. е. не содержащие файлов и поддиректорий (вложенных папок).

Для удаления файлов используется команда rm. Например:

rm ghost.png

Чтобы удалить не пустой каталог можно использовать команду rm с ключом -r. При этом при удаление каждого вложенного объекта будет требоваться подтверждения.

Однако если добавить еще ключ -f, то вопросов на подтвержение уже возникать не будет:

rm -rf /mnt/save/alldocuments/

вопросы

1. Для чего предназначены команды **cp, mv, rm, mkdir, rmdir**?
2. Как скопировать каталог?
3. Можно ли удалить не пустой каталог?
4. В каком из приведенных ниже примерах происходит перемещение файла? его переименование? одновременно оба действия?

а) mv ./work/tech/comp.png ./work/tech/my_car.png

б) mv ./work/tech/comp.png ./Desktop

в) mv ./work/tech/comp.png ./work/tech/computer.png

практическая работа

1. В домашней директории создайте каталог **folder** и переместитесь в него.
2. Создайте в нем три файла любыми известными вам способами, а также вложенный каталог **inside**.

3. Скопируйте один файл из каталога **folder** в каталог **inside**, а два оставшихся — переместите в **inside**. Файл, оставшийся в каталоге folder, переименуйте.
4. Создайте копию каталога **inside** на рабочем столе (**Desktop**).
5. Удалите каталог **folder**.

Виды ссылок в Linux. Теория

что такое индексный дескриптор?

Мы знаем, что файл – это область данных на диске, которую можно найти по имени. Однако в операционных системах на базе ядра Linux вся информация о файле привязана не к имени, а так называемому **индексному дескриптору**. У каждого файла есть свой уникальный (единственный и неповторимый) индексный дескриптор, который содержит сведения о файле: в каких блоках диска хранится содержимое файла, размер файла, время его создания и др.

Пронумерованные индексные дескрипторы файлов содержатся в специальной таблице. Каждый логический и физический диск имеет собственную таблицу индексных дескрипторов.

Именно **номер индексного дескриптора является истинным именем файла в системе**.

Какие ссылки бывают жесткими?

Поскольку индексные дескрипторы представляют собой номера, а файлов в операционной системе обычно очень много, то искать файл по номеру его дескриптора очень неудобно: человеку работать с осмысленными словами куда удобнее, чем с огромными числами. Поэтому любому файлу в системе обычно дается осмысленное имя (обычно словесное), которое не содержит информации о файле, а лишь указывает (ссылается) на его дескриптор.

Имя файла, ссылающееся на его индексный дескриптор, называется жесткой ссылкой. Механизм жестких ссылок – это основной способ обращаться к файлу по имени в операционных системах, основанных на ядре Linux.

сколько имен у файла?

Файл в системе идентифицирует (определяет) номер его индексного дескриптора, а имя файла содержит лишь указатель на него. Естественно, что таких указателей можно создать множество, хотя все они будут направлять на один объект. Для образного сравнения, можно представить придорожные указатели на какую-нибудь бензоколонку: их много, они находятся в разных местах трассы, но указывают на одну и ту же точку. Другими словами, у файла в Linux может быть несколько имен.

Но почему бывает недостаточно одного имени файла? Все дело в удобстве доступа (а также предоставлении доступа).

Например, человек работает над проектом и постоянно обращается к файлу, местоположение которого предусмотрено во вложенном каталоге. Чтобы открыть этот файл в графическом режиме, придется по крайней мере открыть последовательно два каталога. Но куда удобнее будет поместить еще одно имя файла на Desktop (Рабочий стол).

Или предположим такую ситуацию. Администратор системы (человек, управляющий операционной системой) в одном каталоге создал файл, доступ к которому должен быть обеспечен и рядовому пользователю. Однако открывать этот каталог пользователь не имеет права, а переместить файл в каталог пользователя тоже нельзя, т.к он необходим и в данном каталоге. В такой ситуации единственным выходом будет создать еще одну жесткую ссылку на файл и поместить ее в каталог пользователя.

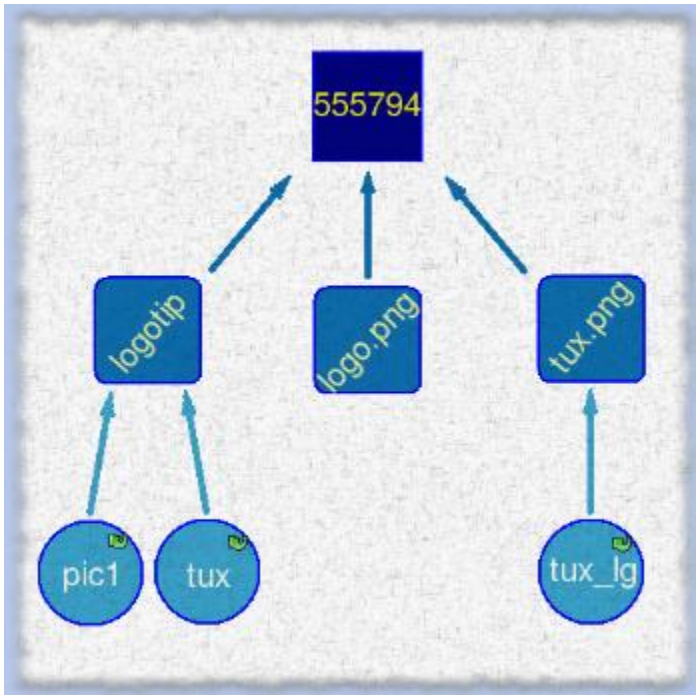
Также, из механизма жестких ссылок вытекает и следующее утверждение: удаление одной жесткой ссылки на файл не приводит к его удалению их системы при наличии у него других жестких ссылок (имен). И это понятно, если учесть, что все жесткие ссылки равны между собой, независимо от времени создания, местонахождения в структуре каталогов и др. Файл будет доступен системе, пока будет существовать хоть одна жесткая ссылка на него. В случае удаления всех ссылок, файл удалится из системы, т.к. станет просто недоступен ей.

в чем "мягкость" мягких ссылок?

Несмотря на всю прелесть жестких ссылок, у них есть и ограничения: их можно создавать только на файлы, но не на каталоги. Также жесткую ссылку нельзя создать с одного диска на другой. Последнее означает, что нельзя создать жесткую ссылку на файл находящийся, например, на съемном носителе (дискеты, флэш-память, CD-R и др.) или другом разделе жесткого диска.

В такой ситуации на помощь приходят мягкие ссылки. Часто их также называют **символьными ссылками**. Они представляют собой файлы, указывающие не на индексные дескрипторы, а на имена файлов.

Следует понимать, что понятия жесткой и мягкой ссылки, несмотря на их созвучность, различны по сути. Жесткая ссылка указывает непосредственно на индексный дескриптор, а мягкая указывает на жесткую ссылку. Если удалить все жесткие ссылки файла, то ни одна мягкая ссылка "не сработает".



На рисунке изображена односторонняя связь между символьными ссылками и именами файла, а также между именами и индексным дескриптором. Верхний квадрат соответствует индексному дескриптору файла, квадраты со скругленными углами – именам файла, а круги – символьным ссылкам. Индексный дескриптор файла всегда один, а имен может быть множество. Также может существовать неограниченное количество символьных ссылок на каждое имя файла. При удалении жесткой ссылки, на имя которой имелась мягкая ссылка, последняя не наследует связь с дескриптором и утрачивает свою "работоспособность". Если в данном примере удалить жесткую ссылку с именем logotip, то файлы pic1 и tux станут бесполезны, т.к. открыть файл 555795 с их помощью уже будет нельзя. Такие ссылки часто называют "битыми".

ссылка – это не копия!

Кому-то может показаться, что копирование файлов и создание ссылок почти одно и то же, т.к. в итоге получаются вроде бы два файла. Однако это абсолютно разные операции, приводящие к совершенно разным результатам.

При копировании файла создается новый файл, данные которого записываются в свободное место на диске, и который имеет собственный индексный дескриптор. В случае же создания жесткой ссылки, файл по-прежнему остается в единственном числе, появляется лишь дополнительный указатель на него.

Практически это имеет следующие последствия. При внесении изменений в файл, обращение к которому было под одним именем, эти изменения обнаружатся и тогда, когда обращение к файлу произойдет под другим именем. При создании копии файла и последующем изменении данных этой копии, данные первоначального файла не изменятся.

В случае мягких ссылок, хоть и создается новый файл (с собственным индексным дескриптором), но он не содержит данных файла-оригинала, а лишь ссылается на жесткую ссылку.

Создание ссылок в Linux

В командной оболочке Linux для того, чтобы добавить файлу еще одно имя (создать еще одну жесткую ссылку на файл) необходимо выполнить команду **ln** (от "link" - ссылка, связывать). В качестве первого параметра указывается существующее имя файла, второго - имя новой ссылки.

ln file1 file2

В данном случае, в текущем каталоге была создана еще одна жесткая ссылка на файл с именем **file1**. Созданная ссылка находится в том же каталоге, что и первая. Далее можно переместить ссылку в другой каталог при помощи команды **mv**.

Можно сразу указать место назначения ссылки с помощью адреса. Например:

ln list ./Desktop/l_class

ln /root/list /home/vasy/Desktop/l_class

В последнем варианте указаны полные имена исходного и нового имен файла.

Число жестких ссылок на файл (т. е. разных имен файла) можно узнать, выполнив команду **ls** с параметром **-l**, которая построчно выводит на экран подробные сведения о каждом объекте каталога.

```
[sveta@ssh ~]$ ln cp_mv_7.odt Desktop/
[sveta@ssh ~]$ ln cp_mv_7.odt Documents/
[sveta@ssh ~]$ ls -l Desktop/ Documents/
Desktop/:
итого 36
-rwxrwxrwx 3 sveta sveta 34002 Июл 15 22:55 cp_mv_7.odt

Documents/:
итого 40
-rwxrwxrwx 3 sveta sveta 34002 Июл 15 22:55 cp_mv_7.odt
-rw-r--r-- 1 sveta sveta    38 Июл  9 00:36 text
[sveta@ssh ~]$
```

В данном примере создаются две ссылки на файд **cp_mv_7.odt**: одна ссылка помещается в папку **Documents**, а вторая на рабочий стол. Далее выполняется команда **ls** по отношению к этим двум каталогам. Сразу за перечислением прав доступа (**-rwxrwxrwx**) к файлу следует число, которое и обозначает количество жестких ссылок на файл. В данном случае их три (не забывайте про ту, что была исходной). То, что все ссылки указывают на один и тот же файл, говорит идентичность информации о файле.

Символьную ссылку можно создать при помощи команды **ln** с ключом **-s** (от "symbolic"). В качестве первого параметра пишется АБСОЛЮТНЫЙ АДРЕС и имя исходного файла, в качестве второго – адрес и имя мягкой ссылки. Например:

```
[sveta@ssh ~]$ ln -s /home/sveta/Documents/text Desktop/link_text
[sveta@ssh ~]$ ls -l Desktop/
итого 184
-rwxrwxrwx 6 sveta sveta  34002 Июл 15 22:55 cp_mv_7.odt
lrwxrwxrwx 1 sveta sveta     26 Июл 19 22:31 link_text -> /home/sveta/Documents/text
```

В примере создана символьная ссылка на файл **text**, находящийся в каталоге **Documents**. Ссылка размещена на рабочем столе. При просмотре содержимого каталога **Desktop**, мы можем видеть, что файл **link_text** является символьной ссылкой на объект находящийся по адресу **/home/sveta/Documents/text**.

В отличие от жестких ссылок, символьные ссылки можно создавать и на каталоги. В принципе, в этом и можно усмотреть их главное практическое назначение. В примере ниже создается ссылка на каталог, находящийся на другом разделе жесткого диска. После этого доступ к нему можно осуществлять непосредственно с рабочего стола (что намного удобнее).

```
[sveta@ssh ~]$ ln -s /mnt/win/hda8/methodica\&education/программа Desktop/обр_прог
[sveta@ssh ~]$ ls -l Desktop/
итого 716
-rwxrwxrwx 6 sveta sveta  34002 Июл 15 22:55 cp_mv_7.odt
lrwxrwxrwx 1 sveta sveta     26 Июл 19 22:31 link_text -> /home/sveta/Documents/text
lrwxrwxrwx 1 sveta sveta     52 Июл 19 22:36 обр_прог -> /mnt/win/hda8/methodica&education/п
рограмма
```

Практическая работа

1. Создайте файл с текстовыми данными, путем перенаправления результата команды **cat** в файл (**cat > myfile**).
2. Далее скопируйте файл, создайте на него жесткую и символьную ссылки. Все объекты оставьте в том же каталоге, что и файл-оригинал.
3. Выполните команду **ls -l**, затем сделайте выводы о том, какие имена указывают на один и тот же объект, а какие на разные.
4. Создайте символьную ссылку на рабочем столе на любой глубоко вложенный каталог файловой структуры.

Вопросы

1. В чем заключаются различия между жесткой и мягкой ссылками?
2. Сколько имен может быть у файла?
3. Есть ли разница между созданием дополнительной ссылки на файл и его копированием? Если "да", то в чем она заключается?

Выводы

- Вся существенная информация о файле привязана к его индексному дескриптору, который представляет собой номер.
- Обратиться к файлу можно по его имени, связанному с индексным дескриптором.
- Количество имен не ограничено.
- Файл останется доступен операционной системе до тех пор, пока не будут удалены все его имена.
- Символьная (мягкая) ссылка – это особый тип файла, ссылающийся на жесткую ссылку.

Команда chmod. Изменение значений прав доступа к файлам

Пояснение

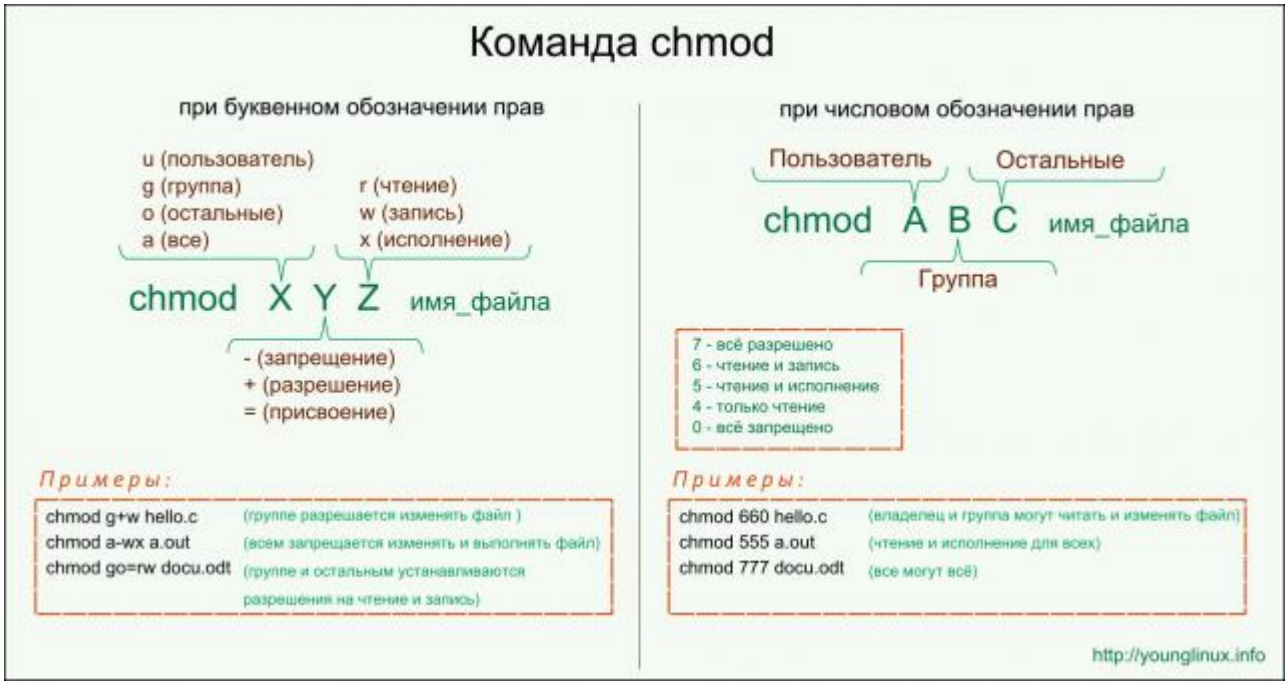
Задача этого урока — изучить, как значения прав на файлы для разных категорий влияют на возможные действия с ним. Придется действовать от имен разных пользователей, поэтому необходимо, чтобы в системе было несколько пользователей.

По умолчанию, когда создается новый пользователь в Linux, он входит только в свою собственную группу. Чтобы увидеть как члены одной группы могут работать с файлом, требуется создать новую группу и добавить туда пару пользователей (или одного пользователя добавить в группу другого).

Поскольку в процессе урока происходит постоянное переключение между пользователями, необходимы элементарные навыки работы в текстовом режиме операционных систем GNU/Linux.

Положительное право исполнять имеет смысл не для всех файлов. Поэтому в уроке предусмотрена подготовка файла с кодом на языке C и его последующая компиляция в бинарный файл.

Для того, чтобы перейти на страницу с описанием команды chmod, щелкните по рисунку ниже.



Команда **chmod** (*change file mode* — сменить режим файла) предназначена для смены/установки значений прав доступа к файлам в Unix-подобных операционных системах.

Синтаксис команды:

команда установка_значений *имя_файла*

Установка значений прав командой chmod может осуществляться двумя способами.

1.

В первом случае права устанавливаются с использованием **трех групп** символов.

В первой группе указывается, кому будет предоставляться или запрещаться доступ: владельцу (**u**), группе (**g**), другим (**o**) или всем (**a**). Можно указать как одну, так сразу несколько категорий.

Вторая группа всегда состоит из одного символа (-, + или =), который обозначает, что конкретно будет делаться со значениями прав (запрещаться, разрешаться или назначаться).

В третьей группе перечисляются права, значения которых подвергаются изменениям: чтение (**r**), запись (**w**), исполнение (**x**). Можно указывать сразу несколько прав.

Примеры использования команды **chmod**:

```
chmod g+w hello.c  (группе разрешается изменять файл)
chmod a-wx a.out  (всем запрещается изменять и выполнять файл)
chmod go=rw docu.odt  (группе и всем остальным устанавливаются разрешения на чтение и запись)
```

2.

Чаще используется второй способ использования команды **chmod**, когда предоставляемые права выражаются одной цифрой для каждой категории граждан. Цифры обозначают следующее:

7 — разрешено чтение, запись и исполнение
6 — разрешены только чтение и запись
5 — разрешены только чтение и исполнение
4 — разрешено только чтение
0 — ничего не разрешено

Почему используются именно такие цифры можно узнать здесь: [числовой способ записи прав доступа к файлам в Linux](#).

Примеры использования команды **chmod**:

```
chmod 660 hello.c  (только владелец и группа могут читать и изменять файл)
chmod 555 a.out  (для всех категорий возможно только чтение и исполнение)
chmod 777 docu.odt  (для всех всё разрешено)
```

Права доступа к файлам в Unix-подобных операционных системах (GNU/Linux)

Опубликовано plustilino в Апрель 2, 2010 - 15:54. Changed Январь 3, 2011 - 18:43

Категории по отношению к файлам

GNU/Linux как истинная Unix-подобная операционная система является многопользовательской. Это значит, что в системе могут работать несколько (или множество) пользователей. Одновременно или по очереди — не важно. Каждый пользователь должен иметь собственное «файловое пространство», к которому доступ других пользователей может быть запрещен или ограничен. Конкретный пользователь по отношению к его собственным файлам выступает как их **владелец (u - user)**.

С другой стороны, очевидно, что в любой системе есть программы и данные, которые должны быть доступны всем пользователям или только определенной группе пользователей. Поэтому у каждого файла есть еще и **группа (g - group)**, к которой он принадлежит.

Файл может быть доступен абсолютно всем. Поэтому в его атрибутах должны содержаться значения, запрещающие или разрешающие доступ всем **другим (o - other)**, кто не вошел в группу и не является владельцем.

Возможные действия над файлом

Что можно делать с файлом после его создания? В первую очередь просматривать, или **читать (r - read)**.

Во вторую очередь, файл можно изменить (дописать, исправить, переименовать, переместить). Таким образом, мы можем говорить о возможности **записи (w - write)** в файл.

Если файл является программой, то его содержимое представляет собой команды для процессора, выполнение которых приводит к тому или иному желаемому (мы надеемся) эффекту. Другими словами, некоторые файлы можно **исполнять (x - execution)**.

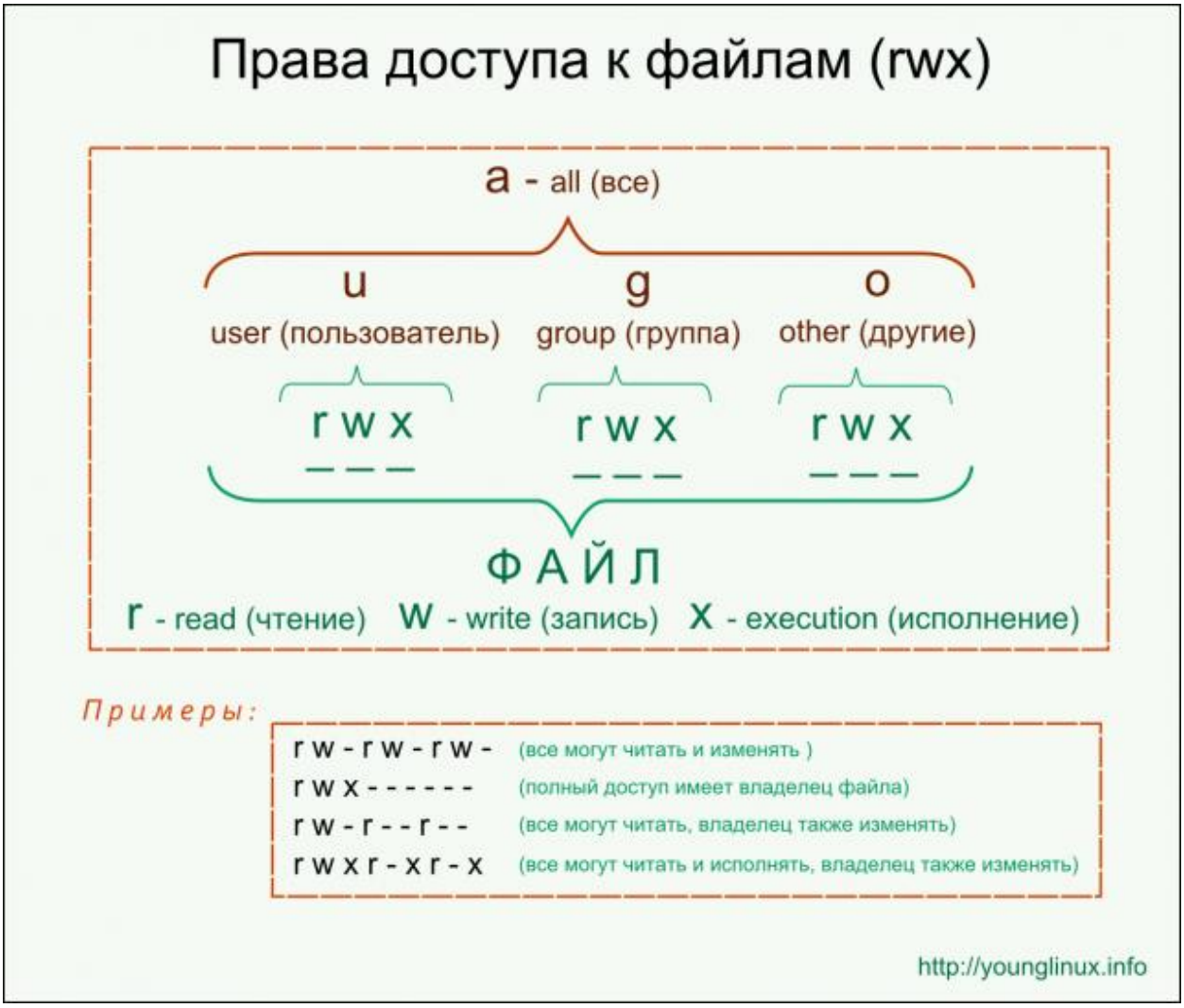
Указание прав доступа с помощью буквенной нотации

Поскольку имеется три категории, претендующие на доступ к файлу (владелец, группа и остальные), и три возможных действия над файлом (чтение, запись и исполнение), то получается, что в атрибутах файла должно быть девять записей о правах, указывающих на то, кто и что может делать с этим файлом. **Первые три записи — это права владельца**, вторые три записи — права группы, *последняя тройка — права по отношению к файлу всех остальных*.

r w x r w x *r w x*

Если значение какого-либо права отрицательно для определенной категории, то вместо буквы ставится тире. Например, в примере ниже, все могут читать файл, но никто не может исполнить, и только один владелец может изменять файл:

r w - r - - *r* - -



Указание прав доступа с помощью числовой нотации

Выразить права доступа к файлу можно не только с помощью букв. Если принять, что положительное значение права (доступ есть) обозначать единицей, а отрицательное (доступа нет) — нулем, то можно получить примерно следующую картину для файла, описанного выше:

1 1 0 1 0 0 *1 0 0*

Именно с помощью нулей и единиц хранятся данные в памяти компьютера, в том числе и права доступа. Под права выделяются 9 бит в атрибутах файла, каждый бит хранит ноль или единицу.

Однако запись из девяти символов достаточно длинная. Чтобы ее сократить используют преобразование двоичных чисел в восьмеричные. Триады нулей и единиц двоичной системы как раз составляют значения от 0 до 7 в восьмеричной системе счисления.

Особенности доступа к каталогам

Каталог — это особый тип файла. Его содержание — это список других файлов.

Каталоги имеют те же «биты прав», что и остальные файлы. Однако то, что эти права означают может быть не таким простым для понимания.

Если каталог можно **читать (r)**, то это означает, что разрешено только узнать список файлов, содержащихся в этом каталоге. Только список файлов, но не их свойства (размер, права доступа и др.).

Если каталог можно **исполнять (x)**, то это означает, что в него можно заходить и просматривать содержимое файлов (доступ к которым разрешен для данной категории), узнавать свойства (атрибуты) файлов. Можно изменить содержимое файла (если его разрешено менять), но не имя файла.

Если каталог можно **изменять (w)**, то это означает, что в нем можно изменять файлы, их имена, удалять их. *Опасность!* Это можно делать даже с файлами, доступ к которым запрещен для данной категории. *Лечение!* Вводят дополнительный t-бит. При его наличии пользователь может изменять только свои файлы.

Следует понимать, что ...

- доступ к конкретному файлу также зависит от наличия доступа на исполнение к каталогам на протяжении всего пути;
- изменять существующие файлы можно, не имея доступа на запись в каталог, достаточно иметь доступ на запись самого файла.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Под **администратором системы** сделать следующее:

1. Создать двух новых пользователей, если в системе их недостаточно. Потребуется три пользователя. Например: sv, pupil7, test.
2. Создать новую группу. Например: real.
   ```
   groupadd real
   ```
3. Добавить двух пользователей в только что созданную группу.
4. ```
   gpasswd -a sv real
   gpasswd -a pupil7 real
   ```
5. Создать общедоступный разделяемый каталог в каталоге /home. Например lesson.
6. ```
   mkdir /home/lesson
   chmod a+wt /home/lesson
   ```
7. Перейти в каталог
   ```
   cd /home/lesson
   ```
8. В каталоге /home/lesson/ создать файл hello.c
   ```
   vim hello.c
   ```

   Нажать клавишу Insert. И ввести программный код:
   ```
   #include <stdio.h>
   main()
   {
        printf("hello world!\n");
   }
   ```

   Нажать Esc. Затем комбинацию :wq (двоеточие и две буквы). Нажать Enter.
9. Скомпилировать файл.
   ```
   gcc hello.c
   ```

   В результате в каталоге должен появиться исполняемый файл a.out
10. Изменить пользователя и группу файлов.

11. `chown sv:real hello.c`
`chown sv:real a.out`

1. Перейдите в текстовый режим (терминал tty2). Войдите в систему под пользователем sv. Перейдите в каталог /home/lesson.
`cd /home/lesson`
2. Сделайте тоже самое для пользователей pupil7 и test (терминалы соответственно tty3 и tty4).
3. Вернитесь к пользователю sv.
4. sv. Просмотрите содержимое каталога:
`ls -l`
Опишите кто является владельцем файлов a.out и hello.c, какой группе они принадлежат.
_____
_____
5. sv. Запретите «остальным» исполнять файл a.out:
`chmod o-x a.out`
6. pupil7. Выполните файл a.out:
`./a.out`
7. test. Попытайтесь выполнить файл a.out также. Что произошло? Объясните разницу между пользователями test и pupil7 по отношению к данному файлу. Что произошло?_____
Причина: _____
8. pupil7. Попытайтесь переименовать файл hello.c:
`mv hello.c hi.c`

Что произошло? _____
Причина: _____
9. pupil7. Попытайтесь установить разрешение на изменение файла hello.c для группы:
`chmod g+w hello.c`

Что произошло? _____
10. sv. Выполните предыдущую команду в роли данного пользователя. Почему пользователь sv может менять значения прав файла hello.c, а pupil7 — нет?
_____
11. pupil7. Переименуйте файл hello.c:
`mv hello.c hi.c`
12. test. Создайте файл:
`cat > filetest`

Напишите несколько строчек, затем нажмите Ctrl + C.
13. test. Просмотрите каталог:
`ls -l`

Просмотрите файл:
`cat filetest`
14. test. Разрешите запись группе и запретите всё для «остальных»:
`chmod 660 filetest`
15. sv. Попытайтесь прочитать файл:
`cat filetest`

Почему отказано в доступе? _____