Лабораторная работа №3

Цель: получить навыки работы с файлами в ОС Linux

Задачи:

Изучить команды работы с файловой системой *Nix систем

Теоретическая часть

Как получить справку

`man` (от англ. manual — руководство) — команда Unix, предназначенная для форматирования и вывода справочных страниц. Поставляется почти со всеми UNIX-подобными дистрибутивами. Каждая страница справки является самостоятельным документом и пишется разработчиками соответствующего программного обеспечения.

Использование

Чтобы вывести справочное руководство по какой-либо команде (или программе, предусматривающей возможность запуска из терминала), можно в консоли ввести:

```
man <command_name>
```

Например, чтобы посмотреть справку по команде ls, нужно ввести man ls.

Для навигации в справочной системе man можно использовать клавиши ↑ и ↓ для построчного перехода, PgUp и PgDn для постраничного перехода вверх и вниз соответственно.

При просмотре больших страниц удобно воспользоваться поиском, для чего следует нажать /, затем набрать строку поиска (и слеш, и строка поиска отобразятся в нижней части экрана) и нажать ↵ Enter. Обратным поиском (снизу вверх) можно воспользоваться, нажав кнопку ?. При этом подсветятся все совпадения с заданным регулярным выражением, и экран прокрутится до первого из них. Для перехода к следующему подсвеченному вхождению нужно нажать n (Next — следующий), либо оставить строку поиска пустой (/, затем ↵ Enter). Для показа предыдущего совпадения надо также использовать вопросительный знак или же нажимать N (заглавную, то есть ⇧ Shift+N).

Для получения краткой справки по командам и горячим клавишам справочной системы нужно нажать H (Help — помощь).

Для выхода из справочной системы используется клавиша Q (Quit — выход).

Для получения детальной инструкции по использованию команды используется конструкция

```
man man
```

Разделы

Справочные страницы поделены на 8 стандартных разделов и один дополнительный. Каждый из разделов соответствует той или иной тематике в рамках установленной операционной системы.

Раздел	Краткое описание
1	Прикладные программы и команды оболочки
2	Системные вызовы ядра (функции языка Си)
3	Библиотечные вызовы (функции различных библиотек, установленных в систему)
4	Специальные файлы (находящиеся обычно в каталоге /dev)
5	Форматы файлов и соглашения
6	Игры
7	Различные описания, соглашения и прочее
8	Команды администрирования системы, которые обычно запускаются от имени суперпользователя
9	Ядро операционной системы (нестандартный раздел)

Номер раздела в команде man указывается вторым аргументом, перед названием справочной страницы. Если номер раздела опущен, то поиск справочной страницы ведётся по всем разделам по порядку.

```
man passwd #раздел 1
man 1 passwd #раздел 1
man 5 passwd #раздел 5
```

Файловая система Linux

Операционные системы хранят данные на диске при помощи **файловых систем**. Классическая файловая система представляет данные в виде вложенных друг в друга **каталогов** (их ещё называют директориями, папками), в которых содержатся **файлы.** Один из каталогов является «вершиной» файловой системы (а выражаясь технически — «корнем», поскольку он служит корнем дерева файловой системе), в нём содержатся (или, если угодно, из него растут) все остальные каталоги и файлы.

Если жёсткий диск разбит на разделы, то на *каждом* разделе организуется отдельная файловая система с собственным корнем и структурой каталогов (ведь разделы полностью изолированы друг от друга).

В Linux корневой каталог называется весьма лаконично — “/”. Полные имена (пути) всех остальных каталогов получаются из “/”, к которому дописываются справа имена последовательно вложенных друг в друга каталогов. Имена каталогов в пути также разделяются символом “/” («слэш»). Например, запись `/home` обозначает каталог “`home`” в корневом каталоге (“/”), а `/home/user` — каталог “`user`” в каталоге “`home`” (который, в свою очередь, в корневом каталоге). Похожий способ записи полного пути используется в системах DOS и Windows, с той разницей, что корневой каталог обозначается литерой устройства с последующим двоеточием, а в качестве разделителя используется символ “\” («обратный слэш»). Перечисленные таким образом каталоги, завершающиеся именем файла, составляют **полный путь** к файлу.

Относительный путь строится точно так же, как и полный — перечислением через “/” всех названий каталогов, встретившихся при движении к искомому каталогу или файлу. Между полным путём и относительным 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.

Монтирование

Корневой каталог в Linux всегда только *один*, а все остальные каталоги в него вложены, т. е. для пользователя файловая система представляет собой единое целое (это отличается от технологии, применяемой в Windows или Amiga, где для каждого устройства, на котором есть файловая система, используется свой корневой каталог, обозначенный литерой, например: “a”, “c”, “d” и т. д.). В действительности, разные части файловой системы могут находиться на совершенно разных устройствах: разных разделах жёсткого диска, на разнообразных съёмных носителях (лазерных дисках, дискетах, флэш-картах), даже на других компьютерах (с доступом через сеть). Для того, чтобы соорудить из этого хозяйства единое дерево с одним корнем, используется процедура **монтирования.**

Монтирование — это подключение в один из каталогов целой файловой системы, находящейся где-то на другом устройстве. Эту операцию можно представить как «прививание» ветки к дереву. Для монтирования необходим пустой каталог — он называется **точкой монтирования.** Точкой монтирования может служить любой каталог, никаких ограничений на этот счёт в Linux нет. При помощи специальной команды (`mount`) мы объявляем, что в данном *каталоге* (пока пустом) нужно отображать файловую систему,

доступную на таком-то *устройстве* или же по сети. После этой операции в каталоге (точке монтирования) появятся все те файлы и каталоги, которые находятся на соответствующем устройстве. В результате пользователь может даже и не знать, на каком устройстве какие файлы располагаются.

Подключённую таким образом («смонтированную») файловую систему можно в любой момент отключить — **размонтировать** (для этого имеется специальная команда `umount`), после чего тот каталог, куда она была смонтирована, снова окажется пустым.

Для Linux самой важной является **корневая файловая система** (root filesystem). Именно к ней затем будут подключаться (монтироваться) все остальные файловые системы на других устройствах. Обратите внимание, что корневая файловая система тоже монтируется, но только не к другой файловой системе, а к «самой Linux», причём точкой монтирования служит "`/`" (корневой каталог). Поэтому при загрузке системы прежде всего монтируется корневая файловая система, а при останове она размонтируется (в последнюю очередь).

Пользователю обычно не требуется выполнять монтирование и размонтирование вручную: при загрузке системы будут смонтированы все устройства, на которых хранятся части файловой системы, а при останове (перед выключением) системы все они будут размонтированы. Файловые системы на съёмных носителях (лазерных дисках, дискетах и пр.) также монтируются и размонтируются автоматически — либо при подключении носителя, либо при обращении к соответствующему каталогу.

Стандартные каталоги

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»). Стандарт **FHS** регламентирует не только перечисленные каталоги, но и их подкаталоги, а иногда даже приводит список конкретных файлов, которые должны присутствовать в определённых каталогах. Краткое описание стандартной иерархии каталогов Linux можно получить, отдав команду `man hier`. Полный текст и последнюю редакцию стандарта **FHS** можно найти в пакете fhs или прочесть по адресу `http://www.pathname.com/fhs/`). Этот стандарт довольно последовательно соблюдается во всех Linux-системах: так, в любой Linux вы всегда найдёте каталоги `/etc`, `/home`, `/usr/bin` и т. п. и сможете довольно точно предсказать, что именно в них находится.

Стандартное размещение файлов позволяет и человеку, и даже программе предсказать, где находится тот или иной компонент системы. Для человека это означает,

что он сможет быстро сориентироваться в любой системе Linux (где файловая система организована в соответствии со стандартом) и найти то, что ему нужно. Для программ стандартное расположение файлов — это возможность организации автоматического взаимодействия между разными компонентами системы.

. – текущий каталог

.. – каталог на уровень выше

/ – корневой каталог

~ – домашний каталог пользователя

Параметры монтирования

При выполнении операции монтирования, в том числе при выборе точки монтирования во время инсталляции Linux-системы, можно изменять свойства смонтированной файловой системы. Для этого нужно указать утилите `mount` один или несколько параметров. Существует ряд параметров монтирования, поддерживаемых всеми файловыми системами. Есть параметры, характерные для одной конкретной файловой системы. Подробно о параметрах монтирования можно прочитать в руководстве к утилите `mount` (`mount(8)`).

Структура каталогов Linux

/ — корень

/bin — (binaries) бинарные файлы пользователя

/sbin — (system binaries) системные исполняемые файлы

/etc — (etcetera) конфигурационные файлы

/dev — (devices) файлы устройств

/proc — (proccess) информация о процессах

/var (variable) — Переменные файлы

/var/log — Файлы логов

/var/lib — базы данных

/var/mail — почта

/var/spool — принтер

/var/lock — файлы блокировок

/var/run — PID процессов

/tmp (temp) — Временные файлы

/usr — (user applications) Программы пользователя

/usr/bin/ — Исполняемые файлы

/usr/sbin/

/usr/lib/ — Библиотеки

/usr/local — Файлы пользователя

/home — Домашняя папка

/boot — Файлы загрузчика

/lib (library) — Системные библиотеки

/opt (Optional applications) — Дополнительные программы

/mnt (mount) — Монтирование

/media — Съемные носители

/srv (server) — Сервер

/run — процессы

/sys (system) — Информация о системе

/ — КОРЕНЬ

Это главный каталог в системе Linux. По сути, это и есть файловая система Linux. Здесь нет дисков или чего-то подобного, как в Windows. Вместо этого, адреса всех файлов начинаются с корня, а дополнительные разделы, флешки или оптические диски подключаются в папки корневого каталога.

Только пользователь root имеет право читать и изменять файлы в этом каталоге.

Обратите внимание, что у пользователя root домашний каталог /root, но не сам /.

/BIN — (BINARIES) БИНАРНЫЕ ФАЙЛЫ ПОЛЬЗОВАТЕЛЯ

Этот каталог содержит исполняемые файлы. Здесь расположены программы, которые можно использовать в однопользовательском режиме или режиме восстановления. Одним словом, те утилиты, которые могут использоваться пока еще не подключен каталог /usr/. Это такие общие команды, как cat, ls, tail, ps и т д.

/SBIN — (SYSTEM BINARIES) СИСТЕМНЫЕ ИСПОЛНЯЕМЫЕ ФАЙЛЫ

Так же как и /bin, содержит двоичные исполняемые файлы, которые доступны на ранних этапах загрузки, когда не примонтирован каталог /usr. Но здесь находятся программы, которые можно выполнять только с правами суперпользователя. Это разные утилиты для обслуживания системы. Например, iptables, reboot, fdisk, ifconfig,swapon и т д.

/ETC — (ETCETERA) КОНФИГУРАЦИОННЫЕ ФАЙЛЫ

В этой папке содержатся конфигурационные файлы всех программ, установленных в системе.

Кроме конфигурационных файлов, в системе инициализации Init Scripts, здесь находятся скрипты запуска и завершения системных демонов, монтирования файловых систем и автозагрузки программ. Структура каталогов linux в этой папке может быть немного запутанной, но предназначение всех их — настройка и конфигурация.

/DEV — (DEVICES) ФАЙЛЫ УСТРОЙСТВ

В Linux все, в том числе внешние устройства являются файлами. Таким образом, все подключенные флешки, клавиатуры, микрофоны, камеры — это просто файлы в каталоге /dev/. Этот каталог содержит не совсем обычную файловую систему. Структура файловой системы Linux и содержащиеся в папке /dev файлы инициализируются при загрузке системы, сервисом udev. Выполняется сканирование всех подключенных устройств и создание для них специальных файлов. Это такие устройства, как: /dev/sda, /dev/sr0, /dev/tty1, /dev/usbmon0 и т д.

/PROC — (PROCCESS) ИНФОРМАЦИЯ О ПРОЦЕССАХ

Это тоже необычная файловая система, а подсистема, динамически создаваемая ядром. Здесь содержится вся информация о запущенных процессах в реальном времени. По сути, это псевдофайловая система, содержащая подробную информацию о каждом процессе, его Pid, имя исполняемого файла, параметры запуска, доступ к оперативной памяти и так далее. Также здесь можно найти информацию об использовании системных ресурсов, например, /proc/cpuinfo, /proc/meminfo или /proc/uptime. Кроме файлов в этом каталоге есть большая структура папок linux, из которых можно узнать достаточно много информации о системе.

/VAR (VARIABLE) — ПЕРЕМЕННЫЕ ФАЙЛЫ

Название каталога /var говорит само за себя, он должен содержать файлы, которые часто изменяются. Размер этих файлов постоянно увеличивается. Здесь содержатся файлы системных журналов, различные кеши, базы данных и так далее. Дальше рассмотрим назначение каталогов Linux в папке /var/.

/VAR/LOG — ФАЙЛЫ ЛОГОВ

Здесь содержатся большинство файлов логов всех программ, установленных в операционной системе. У многих программ есть свои подкаталоги в этой папке, например, /var/log/apache — логи веб-сервера, /var/log/squid — файлы журналов кеширующего сервера squid. Если в системе что-либо сломалось, скорее всего, ответы вы найдете здесь.

/VAR/LIB — БАЗЫ ДАННЫХ

Еще один тип изменяемых файлов — это файлы баз данных, пакеты, сохраненные пакетным менеджером и т д.

/VAR/MAIL — ПОЧТА

В эту папку почтовый сервер складывает все полученные или отправленные электронные письма, здесь же могут находиться его логи и файлы конфигурации.

/VAR/SPOOL — ПРИНТЕР

Изначально, эта папка отвечала за очереди печати на принтере и работу набора программ cpus.

/VAR/LOCK — ФАЙЛЫ БЛОКИРОВОК

Здесь находятся файлы блокировок. Эти файлы означают, что определенный ресурс, файл или устройство занят и не может быть использован другим процессом. Apt-get, например, блокирует свою базу данных, чтобы другие программы не могли ее использовать, пока программа с ней работает.

/VAR/RUN — PID ПРОЦЕССОВ

Содержит файлы с PID процессов, которые могут быть использованы, для взаимодействия между программами. В отличие от каталога /run данные сохраняются после перезагрузки.

/TMP (TEMP) — ВРЕМЕННЫЕ ФАЙЛЫ

В этом каталоге содержатся временные файлы, созданные системой, любыми программами или пользователями. Все пользователи имеют право записи в эту директорию.

Файлы удаляются при каждой перезагрузке. Аналогом Windows является папка Windows\Temp, здесь тоже хранятся все временные файлы.

/USR — (USER APPLICATIONS) ПРОГРАММЫ ПОЛЬЗОВАТЕЛЯ

Это самый большой каталог с большим количеством функций. Тут наиболее большая структура каталогов Linux. Здесь находятся исполняемые файлы, исходники программ, различные ресурсы приложений, картинки, музыку и документацию.

/USR/BIN/ — ИСПОЛНЯЕМЫЕ ФАЙЛЫ

Содержит исполняемые файлы различных программ, которые не нужны на первых этапах загрузки системы, например, музыкальные плееры, графические редакторы, браузеры и так далее.

/USR/SBIN/

Содержит двоичные файлы программ для системного администрирования, которые нужно выполнять с правами суперпользователя. Например, таких как Gparted, sshd, useradd, userdel и т д.

/USR/LIB/ — БИБЛИОТЕКИ

Содержит библиотеки для программ из /usr/bin или /usr/sbin.

/USR/LOCAL — ФАЙЛЫ ПОЛЬЗОВАТЕЛЯ

Содержит файлы программ, библиотек, и настроек созданные пользователем. Например, здесь могут храниться программы собранные и установленные из исходников и скрипты, написанные вручную.

/HOME — ДОМАШНЯЯ ПАПКА

В этой папке хранятся домашние каталоги всех пользователей. В них они могут хранить свои личные файлы, настройки программ и т д. Например, /home/sergiy и т д. Если

сравнивать с Windows, то это ваша папка пользователя на диске C, но в отличии от WIndows, home как правило размещается на отдельном разделе, поэтому при переустановке системы все ваши данные и настройки программ сохранятся.

/BOOT — ФАЙЛЫ ЗАГРУЗЧИКА

Содержит все файлы, связанные с загрузчиком системы. Это ядро vmlinuz, образ initrd, а также файлы загрузчика, находящие в каталоге /boot/grub.

/LIB (LIBRARY) — СИСТЕМНЫЕ БИБЛИОТЕКИ

Содержит файлы системных библиотек, которые используются исполняемыми файлами в каталогах /bin и /sbin.

Библиотеки имеют имена файлов с расширением *.so и начинаются с префикса lib*. Например, libncurses.so.5.7. Папка /lib64 в 64 битных системах содержит 64 битные версии библиотек из /lib. Эту папку можно сравнить с WIndows\system32, там тоже сгружены все библиотеки системы, только там они лежат смешанные с исполняемыми файлами, а здесь все отдельно.

/OPT (OPTIONAL APPLICATIONS) — ДОПОЛНИТЕЛЬНЫЕ ПРОГРАММЫ

В эту папку устанавливаются проприетарные программы, игры или драйвера. Это программы созданные в виде отдельных исполняемых файлов самими производителями. Такие программы устанавливаются в под-каталоги /opt/, они очень похожи на программы Windows, все исполняемые файлы, библиотеки и файлы конфигурации находятся в одной папке.

/MNT (MOUNT) — МОНТИРОВАНИЕ

В этот каталог системные администраторы могут монтировать внешние или дополнительные файловые системы.

/MEDIA — СЪЕМНЫЕ НОСИТЕЛИ

В этот каталог система монтирует все подключаемые внешние накопители — USB флешки, оптические диски и другие носители информации.

/SRV (SERVER) — СЕРВЕР

В этом каталоге содержатся файлы серверов и сервисов. Например, могут содержаться файлы веб-сервера apache.

/RUN — ПРОЦЕССЫ

Еще один каталог, содержащий PID файлы процессов, похожий на /var/run, но в отличие от него, он размещен в TMPFS, а поэтому после перезагрузки все файлы теряются.

/SYS (SYSTEM) — ИНФОРМАЦИЯ О СИСТЕМЕ

Назначение каталогов Linux из этой папки — получение информации о системе непосредственно от ядра. Это еще одна файловая система организуемая ядром и

позволяющая просматривать и изменить многие параметры работы системы, например, работу swap, контролировать вентиляторы и многое другое.

Ссылки

Символическая («мягкая») ссылка (также «симлинк», от англ. Symbolic link) — специальный файл в файловой системе, в котором вместо пользовательских данных содержится путь к файлу, открываемому при обращении к данной ссылке (файлу).

Целью ссылки может быть любой объект: например другая ссылка, файл, каталог или даже несуществующий файл (в последнем случае при попытке открыть его должно выдаваться сообщение об отсутствии файла). Ссылка, указывающая на несуществующий файл, называется висячей или битой.

Символические ссылки используются для более удобной организации структуры файлов на компьютере, так как:

- позволяют для одного файла или каталога иметь несколько имён и различных атрибутов;
- свободны от некоторых ограничений, присущих жёстким ссылкам (последние действуют только в пределах одной файловой системы (одного раздела) и не могут ссылаться на каталоги).

```
ln -s файл имя_ссылки
# Создаётся «символическая ссылка (symbolic link)
```

Даже если при создании символической ссылки (используя ключ «-s») обозначаемый «файл» окажется несуществующим, символическая ссылка всё равно будет создана (с именем «имя_ссылки»

Жёсткой ссылкой (англ. hard link) в UFS-совместимых файловых системах называется структурная составляющая файла — описывающий его элемент каталога.

Файл в UFS представляет собой структуру блоков данных на диске, имеющую уникальный индексный дескриптор (или i-node) и набор атрибутов (метаинформацию). Жёсткая ссылка связывает индексный дескриптор файла с каталогом и дает ему имя.

Свойства

- У файла может быть несколько жёстких ссылок: в таком случае он будет фигурировать на диске одновременно в различных каталогах или под различными именами в одном каталоге. При редактировании файла через одну из ссылок на него, содержимое по другим ссылкам тоже изменится.
- Количество жёстких ссылок файла сохраняется на уровне файловой системы в метаинформации. Файлы с нулевым количеством ссылок перестают существовать для системы и, со временем, будут перезаписаны физически. В

файловых системах UNIX-подобных ОС и в NTFS при создании файла на него автоматически создаётся одна жёсткая ссылка (на то место файловой системы, в котором файл создаётся). Дополнительную ссылку в UNIX можно создать с помощью команды `ln`. Все ссылки одного файла равноправны и неотличимы друг от друга — нельзя сказать, что файл существует в таком-то каталоге, а в других местах есть лишь их копии. Удаление ссылки приводит к удалению файла лишь в том случае, когда это была последняя ссылка, любая из созданных, то есть все остальные жёсткие ссылки на него уже удалены

- Большинство программ не различают жёсткие ссылки одного файла, даже системный вызов для удаления файла в UNIX называется unlink, так как он предназначен для удаления жёсткой ссылки файла.
- В связи с тем, что жёсткие ссылки ссылаются на индексный дескриптор, уникальный в пределах дискового раздела, создание жёсткой ссылки на файл в каталоге другого раздела невозможно. Для преодоления этого ограничения используются символьные (символические) ссылки.

Работа с файлами и каталогами

Текущий каталог

Файловая система не только систематизирует данные, но и является основой метафоры «рабочего места» в Linux. Каждая выполняемая программа «работает» в строго определённом каталоге файловой системы. Такой каталог называется **текущим каталогом**, можно представлять, что программа во время работы «находится» именно в этом каталоге, это её «рабочее место». В зависимости от текущего каталога может меняться поведение программы: 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 взаимодействие пользователя с системой обязательно опосредовано командной оболочкой, можно говорить о том, что пользователь «находится» в том каталоге, который в данный момент является *текущим каталогом его командной оболочки*.

Все команды, отдаваемые 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`:

`pwd` (аббревиатура от **p**rint **w**orking **d**irectory) возвращает полный путь текущего каталога командной оболочки, естественно, именно той командной оболочки, при помощи

которой была выполнена команда `pwd`. В данном примере текущим является каталог «`/home/methody`».

Текущий каталог, каков бы ни был полный путь к нему, всегда имеет ещё одно обозначение, «`.`», которое можно использовать, если по каким-то причинам требуется, чтобы даже в *относительном* пути к файлу, находящемуся в *текущем* каталоге, присутствовал элемент «имя каталога». Так, пути «`text`» и «`./text`» тоже приводят к одному и тому же файлу, однако в первом случае в строке пути не содержится ничего, кроме имени файла.

Отделить путь к файлу от его имени можно с помощью команд `dirname` и `basebane` соответственно:

Мефодий заметил, что для «`text`» и «`./text`» `dirname` выдало одинаковый результат: «.», что понятно: как было сказано выше, эти формы пути совершенно эквивалентны, а при *автоматической* обработке результатов `dirname` гораздо лучше получить «.», чем *пустую строку*.

Информация о каталоге

В любой момент можно просмотреть содержимое любого каталога при помощи утилиты `ls` (сокращение от англ. «list» — «список»):

Поданная без параметров, команда `ls` выводит список файлов и каталогов, содержащихся в *текущем каталоге*[1].

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` «понимает» множество ключей, которые нужны главным образом для того, чтобы выводить дополнительную информацию о файлах в каталоге или выводить список файлов выборочно. Чтобы узнать обо всех возможностях `ls`, нужно, конечно же, прочесть **руководство** по этой утилите («`man ls`»).

При наличии этого ключа F `ls` в конце имени каждого каталога ставит символ «/», чтобы показать, что в нём может содержаться что-то ещё. В выведенном списке нет ни одного файла — в корневом каталоге содержатся только подкаталоги.

Внезапно обнаружилось, что файлов в домашнем каталоге не два, а гораздо больше. Дело в том, что утилита `ls` по умолчанию не выводит информацию об объектах, чьё имя начинается с «`.`» — в том числе о «`.`» и «`..`». Для того, чтобы посмотреть *полный* список содержимого каталога, и используется ключ «`-a`» (**a**ll). Как правило, с «`.`» начинаются имена **конфигурационных файлов** и **конфигурационных каталогов**.

«`..`» — это ссылка на **родительский каталог**. Родительский каталог — это тот каталог, *в котором* находится данный. Родительским каталогом для «`/home/methody`» будет каталог «`/home`»: он получается просто отбрасыванием последнего имени каталога в полном пути. Иначе можно сказать, что родительский каталог — это один шаг по дереву каталогов по направлению к корню. «`..`» — это сокращённый способ сослаться на родительский каталог: пока текущим каталогом является «`/home/methody`», **относительный путь** «`..`» (или, что то же самое, «`./..`») будет эквивалентен «`/home`». С использованием «`..`» можно строить *сколь угодно* длинные пути, такие как «`../../usr/../var/log/../run/../../home`». 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 имён: «`.`» и «`..`».

Перемещение по дереву каталогов

Пользователь может работать с файлами не только в своём домашнем каталоге, но и в других каталогах. В этом случае будет удобно *сменить* **текущий каталог**, т. е. «переместиться» в другую точку файловой системы. 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. Как обычно, в качестве имени каталога можно использовать полный или относительный путь.

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 заменён 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, эквивалента команде «`cd ~`» и делает текущим каталогом домашний каталог пользователя.

Создание каталогов

Для этого используется утилита `mkdir`. Она используется с одним обязательным параметром: именем создаваемого каталога. По умолчанию каталог будет создан в текущем каталоге.

Копирование и перемещение файлов

Для перемещения файлов и каталогов предназначена утилита `mv` (сокращение 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, изменяются *каталоги,* в которых произошло перемещение. Перемещение предполагает удаление ссылки на файл из того каталога, откуда он перемещён, и добавление ссылки на этот самый файл в тот каталог, куда он перемещён. 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` (сокращение от англ. «copy» — «копировать»). Утилита `cp` требует присутствия 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. Есть несколько возможностей при 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 не выводится никакого предупреждения. В этом случае файл будет просто перезаписан, а данные, которые содержались в старой версии файла, бесповоротно потеряны. Поэтому при использовании `cp` следует всегда быть внимательным и проверять имена файлов, которые нужно скопировать.

Удаление файлов и каталогов

В Linux для удаления файлов предназначена утилита `rm` (сокращение от англ. «remove» — «удалять»).

Однако удалить командой `rm` каталог не получится:

Для удаления *каталогов* предназначена другая утилита — `rmdir` (от англ. «remove directory»). Впрочем, `rmdir` согласится удалить каталог только в том случае, если он пуст: в нём нет никаких файлов и подкаталогов. Удалить каталог вместе со всем его содержимым можно командой `rm` с ключом «`-r`» (**r**ecursive). Команда `rm -r` *`каталог`* — очень

удобный способ потерять в одночасье *все* файлы: она рекурсивно обходит весь *каталог*, удаляя всё, что попадётся: файлы, подкаталоги... а ключ «`-f`» (**f**orce) делает её работу ещё неотвратимее, так как подавляет запросы вида «удалить защищённый от записи файл», так что `rm` работает безмолвно и безостановочно.

В Linux не предусмотрено процедуры восстановления удалённых файлов и каталогов. Поэтому стоит быть *очень* внимательным, отдавая команду `rm` и, тем более, `rm -r`: нет никакой гарантии, что удастся восстановить случайно удалённые данные. Узнав об этом, Мефодий не огорчился, но подумал, что впредь будет удалять только *действительно* ненужные файлы, а всё сомнительное — перемещать с помощью `mv` в подкаталог `~/tmp`, где оно не будет мозолить глаза, и где можно периодически наводить порядок.

Создание файлов

Файл можно создать несколькими способами:

- `touch` — команда Unix, предназначенная для установки времени последнего изменения файла или доступа в текущее время. Также используется для создания пустых файлов;
- `cat > файл` – вообще команда `cat` используется для вывода файла, но с помощью треугольной скобки можно изменить направление вывода (в данном случае на ввод). И в отличии от предыдущего способа можно сразу вводить в файл данные. Завершить процесс ввода можно нажав CTRL + C;
- через любой редактор, например, `vi` или более простой `nano`.

Просмотр содержимого файлов

- `cat файл`. От англ. “concatenate” – утилита UNIX, выводящая последовательно указанные файлы (или устройства), таким образом, объединяя их в единый поток. Если вместо имени файла указывается «-», то читается стандартный ввод;
- `more файл` – выведет файл в постраничном режиме (также `more` можно использовать через конвейер, например, `cat файл | more`);
- `less` — программа для текстовых терминалов UNIX-подобных систем, используемая для просмотра (но не изменения) содержимого текстовых файлов на экране. Отображает файл с возможностью прокрутки. `less` — улучшение утилиты `more`. Возможна и обратная прокрутка. В отличие от многих текстовых редакторов (которые также можно использовать для просмотра файлов), `less` не нуждается в чтении всего файла перед стартом и в результате быстрее работает с большими файлами;

- `head` – утилита в UNIX и UNIX-подобных системах, выводящая первые n строк из файла, по умолчанию n равно 10;
- `tail -` утилита в UNIX и UNIX-подобных системах, выводящая последние n строк из файла, по умолчанию n равно 10;
- через любой редактор, например, `vi` или более простой `nano`.

Практическая часть

В домашней директории создать каталоги, файлы (различными способами), вывести содержимое, создать ссылки разных типов, удалить файлы и каталоги.

1. Создать каталог в домашней директории. Просмотреть его полный путь
2. Вернитесь в домашний каталог и посмотрите информацию о домашнем каталоге
3. Посмотрите полный список содержимого домашнего каталога
4. Вернитесь в ранее созданный каталог
5. Создайте пустой файл и файл с данными, где укажите ФИО и свою группу
6. В текущем каталоге создайте еще два каталога
7. Переместите пустой файл в один из вновь созданных каталогов
8. Файл с данными скопируйте в другой каталог
9. Перейдите в каталог с файлом с данными. Просмотрите содержимое файла
10. В текущем каталоге создайте файл с данными, чтоб содержимое файла превышало 20 строк.
11. Выведите по очереди первые 10 строк и последние 10 строк вновь созданного файла
12. В домашнем каталоге создайте новый каталог, в нем создайте пустой файл, затем удалите последовательно файл и каталог
13. В первом созданном каталоге создайте файл с данными. Создайте несколько жестких ссылок на файл в разных каталогах. Просмотрите количество жестких ссылок.
14. Создайте символьную ссылку на текущий файл в домашнем каталоге
15. Создайте в домашнем каталоге символьную ссылку на первый созданный каталог

Контрольные вопросы

1. Дайте определение термину «файловая система»
2. Приведите пример классификации файловых систем
3. Приведите команду для вывода справочного руководства по какой-либо команде
4. Приведите организацию справочных страниц. Опишите разделы
5. Укажите виды файловых путей и отметьте различия

6. Дайте определение следующим терминам: монтирование, точка монтирования, размонтирование
7. Перечислите основные этапы операции монтирования
8. Перечислите и опишите стандартные каталоги в Linux
9. Дайте определение термину «символическая ссылка»
10. Укажите, для чего используются символические ссылки
11. Дайте определение термину «жесткая ссылка»
12. Перечислите и опишите свойства жестких ссылок
13. Объясните, есть ли разница между созданием дополнительной ссылки на файл и его копированием. В случае положительного ответа укажите, в чем она заключается