**Задание на ПЗ № 03**

***ПРЕДВАРИТЕЛЬНО*** *(в самом начале):*

* *в домашнем каталоге или* **/tmp/ikpiXX** *(где* **ХХ** *– две последние цифры номера группы) создать каталог* **pz04**
* *в домашнем каталоге (или* **/tmp/ikpiXX/pz04***создать файл отчёта* **Фамилия\_Инициалы.txt** *Все действия пользователя «протоколировать» в файле.   
  В начале вывести на экран, а затем продублировать выводом в файл* **Фамилия\_Инициалы.txt***с помощью команды:* **command\_linux >> Фамилия\_Инициалы.txt**
* *в конце занятия защитить отчёт преподавателю.*

**Методика выполнения задания**

1. Ознакомиться с общими сведениями о файловых системах ОС Linux и структурой файловой системы ОС.
2. Изучить таблицу файловой системы /etc/fstab и используемые параметры.
3. Изучить и законспектировать возможности команды fsck.
4. Структура файловой системы Linux
5. Изучить команды работы с файлами.
6. Используя команды ОС, создать текстовые файла.
7. Полученные два файла объединить в один файл и его содержимое просмотреть на экране.
8. Создать новую директорию и переместить в нее полученные файлы.
9. Вывести полную информацию обо всех файлах и проанализировать уровни доступа.
10. Добавить для всех трех файлов право выполнения членам группы и остальным пользователям.
11. Просмотреть атрибуты файлов.
12. Создать еще один каталог.
13. Установить дополнительную связь объединенного файла с новым каталогом, но под другим именем.
14. Создать символическую связь.
15. Сделать текущим новый каталог и вывести на экран расширенный список информации о его файлах.
16. Произвести поиск заданной последовательности символов в файлах текущей директории и получить перечень соответствующих файлов.
17. Получить информацию об активных процессах и имена других пользователей.
18. Сдать преподавателю отчет о работе и удалить свои файлы и каталоги.
19. Выйти из системы.

**1.  Просмотр файла-таблицы файловой системы /etc/fstab**

Чтобы смонтировать файловые системы во время загрузки, Linux постоянно хранит список файловых систем и их параметров в таблице **/etc/fstab**.

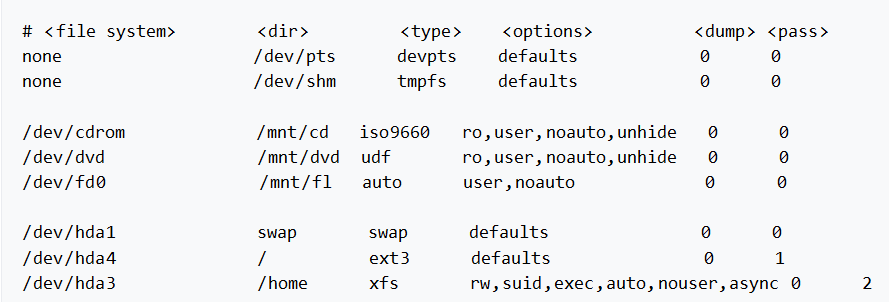
Файл **fstab** (сокр. от англ. **file systems table**) — один из ***конфигурационных файлов*** в UNIX-подобных системах, который содержит информацию о различных файловых системах и устройствах хранения информации компьютера; описывает, как диск (раздел) будет использоваться или как будет интегрирован в систему.

Полный путь к файлу — **/etc/fstab**.

Это файл в обычном текстовом формате, достаточно простом, как можно увидеть из примера 1.

Пример 1. Список файловых систем и их параметров в файле /etc/fstab (*стоит обратить внимание на тот факт, что* **<options>** *могут быть составлены из нескольких значений разделенных запятой*):

**more /etc/fstab**



Каждая строка, содержащая **шесть полей**, соответствует одной файловой системе. Ниже перечислены эти поля (слева направо).

* Устройство или идентификатор **UUID**. Большинство современных систем Linux больше не использует устройство в файле **/etc/fstab**, предпочитая идентификатор **UUID**. Обратите внимание на то, что запись **/proc** содержит устройство-заместитель с именем **proc**.
* Точка монтирования. Указывает, где присоединяется файловая система.
* Тип файловой системы. Скорее всего, вам незнаком параметр **swap** в данном перечне; **это раздел подкачки**.
* Параметры. Использованы длинные параметры, разделенные запятыми.
* Информация о резервной копии для использования командой сброса. В этом поле всегда следует указывать значение 0.
* Порядок проверки целостности системы. Чтобы команда fsck всегда начинала работу с корневого каталога, устанавливайте в этом поле значение 1 для корневой файловой системы и значение 2 для остальных файловых систем на жестком диске. Используйте значение 0, чтобы отключить при запуске проверку чего-либо еще, включая приводы CD-ROM, область подкачки и файловую систему /proc.

**Структура файла fstab**

Каждая запись имеет следующие поля (которые разделяются пробелом или табуляцией):

**<file system> <dir> <type> <options> <dump> <pass>**

1) Поле, **<file system>** (файловая система) сообщает демону монтирования файловых систем **mount**, что монтировать, имя монтируемого устройства.

2) Второе поле, **<dir>** (директория), определяет путь, по которому будет смонтирована **<file system>.**

3) Поле **<type>** (тип) содержит тип файловой системы монтируемого устройства. Полный список поддерживаемых систем можно просмотреть выполнив команду:

**man mount**

Самые распространённые файловые системы: **ext3, ext4, ReiserFS, XFS, JFS, smbfs, ISO9660, VFAT, NTFS, swap**. Ключ **auto** не является файловой системой, он позволяет определять, какой тип файловой системы используется, автоматически. Это удобно для съёмных устройств, дисководов и cd-дисков.

4) Следующее поле называется **<options>** (опции).

Если используются все значения по умолчанию, то используется специальный ключ **defaults**. Если хоть одна опция задана явно, то **defaults** указывать не нужно (**defaults** служит только для того, что была занята позиция в строке). Для полного списка опций используйте команду man **mount**:

|  |  |
| --- | --- |
| **auto** | Файловая система монтируется при загрузке автоматически или после выполнения команды **'mount** -a'. |
| **noauto** | Файловая система может быть смонтирована только вручную. |
| **exec** | Позволяет исполнять бинарные файлы на разделе диска. Установлено по умолчанию. |
| **noexec** | Бинарные файлы не выполняются. Использование опции на корневой системе приведёт к её неработоспособности. |
| **ro** | Монтирует файловую систему только для чтения. |
| **rw** | Монтирует файловую систему для чтения/записи. |
| **sync** | Все операции ввода-вывода должны выполняться синхронно. |
| **async** | Все операции ввода-вывода должны выполняться асинхронно. |
| **user** | Разрешает любому пользователю монтировать файловую систему. Применяет опции **noexec**, **nosuid**, **nodev**, если они не переопределены. |
| **nouser** | Только суперпользователь может монтировать файловую систему. Используется по умолчанию. |
| **defaults** | Использовать значения по умолчанию. Соответствует набору **rw, suid, dev**, **exec, auto, nouser, async**. |
| **suid** | Разрешить операции с **suid** и **sgid** битами. В основном используются, чтобы позволить пользователям выполнять бинарные файлы со временно приобретёнными привилегиями для выполнения определённой задачи. |
| **nosuid** | Запрещает операции с **suid** и **sgid** битами. |
| **nodev** | Данная опция предполагает, что на монтируемой файловой системе не будут созданы файлы устройств **(/dev**). Корневой каталог и целевая директория команды **chroot** всегда должны монтироваться с опцией **dev** или **defaults**. |
| **atime** | Включает запись информации о последнем времени доступа (**atime**) при каждом чтении файла. Включено по умолчанию на Linux до v.2.6.29 включительно. |
| **noatime** | Отключает запись информации о последнем времени доступа (**atime**) при каждом чтении файла. |
| **relatime** | Включает запись информации о последнем времени доступа при чтении файла, если предыдущее время доступа (**atime**) меньше времени изменения файла (**ctime**). Включено по умолчанию на Linux начиная с v.2.6.30. |
| **notail** | Отключает «упаковку хвостов файлов». Опция работает только с файловой системой **ReiserFS**. |

5) **<dump>** - используется утилитой **dump** для того чтобы определить, когда делать резервную копию. После установки, **dump** проверяет эту запись и использует значение, чтобы решить, подключать ли файловую систему. Возможные значения 0 или 1. Если 0, dump игнорирует файловую систему, если 1, **dump** сделает резервную копию. У большинства пользователей **dump** не установлен, поэтому в поле <**dump**> следует задать 0.

6) **<pass>** (номер прохода). **fsck** проверяет число, подставленное в поле <pass> и решает, в каком порядке проверять файловую систему. Возможные значения 0, 1 и 2. Файловые системы со значением <**pass**>, равным 0, не будут проверены утилитой **fsck**. У корневой системы должен быть наибольший приоритет, 1, остальные файловые системы должны иметь приоритет 2.

Вместо указания имени устройства, можно указать **UUID** или метку тома,

написав **LABEL=<label>** или **UUID=<uuid>,**

например, `**LABEL=Boot'** или `**UUID=3e6be9de-8139-11d1-9106-a43f08d823a6'**.

=====================================================================

**2. Проверка и восстановление файловых систем**

*(Пункт не выполнять!!! Прочитать и законспектировать)*

ПРИМЕР ИСПОЛЬЗОВАНИЯ **fsck** ДЛЯ ПРОВЕРКИ ФАЙЛОВЫХ СИСТЕМ

**fsck** - команда UNIX, которая проверяет и устраняет ошибки в файловой системе. **fsck** используется для проверки файловых систем и для исправления ошибок файловой системы, если они будут обнаружены. Основное требование при проверки файловой системы: файловая система должна быть размонтирована. Запуск **fsck** для уже смонтированной файловой системы может привести к ее разрушению. К счастью, при запуске **fsck** на смонтированной файловой системе, программа выдаёт предупреждение. Однажды я его проигнорировал, результат был печальный.

Программа **fsck** может использоваться для проверки всех файловых систем, которые поддерживаются ядром **Linux**.

Формат вызова программы следующий(из под **root**):

**fsck** [**параметры**] [**файловая\_система**]

Параметры и файловую систему, можно не указывать. Если вы не укажете файловую систему, программа начнет проверять все файловые системы, перечисленные в файле **/etc/fstab**. Это очень опасно, так как эти файловые системы могут быть смонтированными, и это может привести к разрушению файловой системы.

Последовательность проверки файловой системы должна быть следующая:

1. Размонтировать файловую систему.

2. Запустить **fsck** для ее проверки.

Например, для проверки файловой системы раздела **/dev/hda1** сначала размонтируем его, а потом запустим **fsck**:

**umount /dev/hda1**

**fsck /dev/hda1**

Но иногда не представляется возможным размонтировать файловую систему, например, когда нужно проверить корневую файловую систему **(/)**. В таком случае необходимо выполнить следующие действия:

Перезагрузиться в однопользовательском режиме.

Перемонтировать корневую файловую систему в режиме "только чтение".

Произвести проверку файловой системы.

Для перезагрузки в однопользовательском режиме перезагрузите систему (команда reboot), а при загрузке передайте ядру параметр **single**.

В однопользовательском режиме может работать только один пользователь — root, а все сервисы выключены, так что проверке файловой системы ничто не будет помешать. Для перемонтирования файловой системы в режим только для чтения введите команду:

**mount -о remount го -t ext3 /**

Параметр **-о** команды **mount** позволяет указать различные опции. В данном случае указана опция **remount** и го, что означает перемонтировать в режиме "только чтение". Параметр **-t** указывает тип файловой системы — **ext3**, а последний параметр — это корневая файловая система (**/**).

Код, возвращаемый **fsck**, является суммой следующих условий:

0 — нет ошибок

1 — ошибки файловой системы исправлены

2 — необходима перезагрузка системы

4 — ошибки файловой системы не исправлены

8 — в процессе проверки произошли ошибки

16 — неверное использование команды либо синтаксическая ошибка

32 — fsck была прервана пользователем

128 — ошибка разделяемых объектов

Примечание.

*Никогда не используйте команду* **fsck** *для смонтированной файловой системы, поскольку ядро может изменить данные на диске во время работы проверки. Это вызовет несоответствия во время исполнения, которые могут привести к сбою системы и повреждению файлов. Есть всего одно исключение: если вы монтируете корневой раздел только для чтения в режиме единственного пользователя, то в этом разделе можно запустить команду* **fsck***.*

=====================================================================

**3. Cтруктура файловой системы Linux**

*(Прочитать и законспектировать)*

Понимание файловой системы Linux, структуры каталогов, размещения конфигурационных, исполняемых и временных файлов поможет вам лучше разбираться в своей системе и стать успешным системным администратором.

Файловая система Linux будет непривычна именно для новичка, только что перешедшего с Windows, ведь здесь все совсем по-другому. В отличие от Windows, программа не находится в одной папке, а, как правило, распределена по корневой файловой системе. Это распределение поддается определенным правилам.

Вы когда-нибудь задавались вопросом, почему некоторые программы находятся в папке /bin, или /sbin, /usr/sbin, /usr/local/bin, в чем разница между этими каталогами?

Например, программа less, находится в каталоге /usr/bin, но почему не в /sbin или /usr/sbin. А такие программы, как ifconfig или fdisk находятся в каталоге /sbin и нигде иначе.

**/** - 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.

**Выводы**

Теперь структура файловой системы Linux не кажется вам темным лесом. Конечно еще есть с чем разбираться, одна только подсистема /proc, /sys или /dev чего стоит. Но база уже заложена. Если я что-то упустил - напишите в комментариях.

=====================================================================

**4. Примеры некоторых команд работы с файлами**

**chmod g+rw,o+r file.1** - установка атрибутов чтения и записи для группы и чтения для

всех остальных пользователей;

**ls -l file.1** - чтение атрибутов файла;

**chmod o-w file.1** - отмена атрибута записи у остальных пользователей;

**>letter** - создание файла letter. Символ > используется как для переадресации, так и для

создания файла (ВНИМАНИЕ: *файл будет создаваться/перезаписываться*

*«с нуля»*);

**>>letter –** добавление в конец файла letter

**cat** - вывод содержимого файла;

**cat file.1 file.2 > file.12** - конкатенация файлов (объединение);

**mv file.1 file.2** - переименование файла file.1 в file.2;

**mv file.1 file.2 file.3 directory** - перемещение файлов file.1, file.2, file.3 в указанную

директорию;

**rm file.1 file.2 file.3** - удаление файлов file.1, file.2, file.3;.

**cp file.1 file.2** - копирование файла с переименованием;

**mkdir namedir** - создание каталога;

**rm dir\_1 dir\_2** - удаление каталогов dir\_1 dir\_2;

**ls [acdfgilqrstv CFR] namedir** - вывод содержимого каталога; если в качестве namedir

указано имя файла, то выдается вся информация

об этом файле.

**cd <namedir>** - переход в другой каталог. Если параметры не указаны, то происходит переход в домашний каталог пользователя.

**pwd** - вывод имени текущего каталога;

**grep [-vcilns] [шаблон поиска] <имя файла>** - поиск файлов с указанием или без указания контекста (шаблона поиска).

Значение ключей:

- v – выводятся строки, не содержащие шаблон поиска;

- c – выводится только число строк, содержащих или не содержащих шаблон;

- i – при поиске не различаются прописные и строчные буквы;

- l – выводятся только имена файлов, содержащие указанный шаблон;

- n – перенумеровать выводимые строки;

- s – формируется только код завершения.

Примеры.

1. Напечатать имена всех файлов текущего каталога, содержащих последовательность "**f**" и имеющих расширение **.txt**:

**grep -l f\*.txt**

1. Определить имя пользователя, входящего в ОС LINUX с терминала tty23:

**who | grep tty23**

=====================================================================

**5. Создать** разными способами текстовые файлы в каталоге **/tmp/ikpiXX/pz04**

Например, файлы: **f1.txt, f2.txt, f3.txt** и т.д.

1) Редактор **Vi / Vim :**

* Здесь есть два режима: **режим вставки** и **командный**.
* Переключаться между ними можно с помощью кнопки **Esc**
* Для выхода из редактора в командном режиме наберите **:q**
* для сохранения файла **:w**

Вообще, **Vim** – очень полезный инструмент. Чтобы узнать побольше о его возможностях и выучить основы выполните: **vimtutor**.

2) **Оператор перенаправления >**

Это, наверное, самая короткая команда для создания файла в Linux:

**$ > файл**

Оператор оболочки для перенаправления вывода позволяет записать вывод любой команды в новый файл. Например, можно подсчитать md5 сумму и создать текстовый файл в linux с результатом выполнения.

**$ md5sum файл > ~/sum.md5**

NB: ***md5summer*** *— программа, позволяющая вычислять значения* ***хеш‐сумм*** *(****контрольных сумм****) файлов по алгоритму MD5. В обычном случае вычисленные хеши выводятся (можно сохранить в файле, для последующего использования). В других случаях, программа сверяет вычисленные значения со значениями, сохраненными в файле (это удобно для массовой проверки целостности файлов).*

*Наиболее часто программа используется для проверки правильной загрузки файлов по сети. Программа имеет множество версий (функциональность версий может существенно различаться) для различных ОС — например, GNU/Linux, UNIX (напр., BSD), Microsoft Windows.*

***MD5*** *(****Message Digest 5****) — 128-битный алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом в 1991 году.*

*Предназначен для создания «отпечатков» или «дайджестов» сообщений произвольной длины.*

***Хеширование*** *(иногда* ***хэширование****, англ.* ***hashing****) — преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины таким образом, чтобы изменение входных данных приводило к непредсказуемому изменению выходных данных. Такие преобразования также называются* ***хеш****-****функциями*** *или* ***функциями******свёртки****, а их результаты называют* ***хешем****,* ***хеш****-****кодом*** *или* ***дайджестом******сообщения*** *(англ.* ***message******digest****).*

3) **Оператор перенаправления >** порождает еще несколько способов создания файла в Linux, например, выведем строку в файл с помощью команды **echo**:

**$ echo "Это строка" > файл.txt**

4) Еще тем же способом можно сделать примитивный текстовый редактор для создания файла. Утилита cat без параметров принимает стандартный ввод, используем это:

**$ cat > файл.txt**

После выполнения команды можете вводить любые символы, которые нужно записать в файл, ***для сохранения нажмите*** **Ctrl+D**

5) А еще есть утилита **printf**, и здесь она тоже поддерживает форматирование вывода:

**$ printf "Это %d текстовая строка\n" 1 > файл**

**cat файл**

Выйдет на экран: **Это 1 текстовая строка**

6) **Оператор перенаправления вывода >>**

Также можно не только перезаписывать файл, а дописывать в него данные, с помощью перенаправления оператора >>. Если файла не существует, будет создан новый, а если существует, то строка запишется в конец.

**$ echo "Это текстовая строка" > файл.txt**

**$ echo "Это вторая текстовая строка" >> файл.txt**

7) **Команда cp**

Команда cp используется для копирования файлов в Linux. Но с помощью нее можно и создать файл. Например, чтобы создать пустой файл можно просто скопировать **/dev/null**:

**$ cp /dev/null файл**

8) **touch**

Вот мы и подобрались к непосредственно созданию файлов через терминал, для этого в Linux есть специальная утилита **touch**. Она позволяет создать пустой файл в Linux, при этом указывать их дату создания, права доступа и другие метаданные.

Чтобы создать пустой файл **linux** просто наберите:

**$ touch файл.txt**

Можно создать несколько пустых файлов сразу:

**$ touch файл1 файл2**

Опция -t позволяет установить дату создания. Дата указывается опцией **-t** в формате **YYMMDDHHMM.SS**. Если не указать будет установлена текущая дата. Пример:

**$ touch -t 201601081830.14 файл**

Можно использовать дату создания другого файла:

**$ touch -r шаблон файл**

Также можно установить дату последней модификации, с помощью опции **-m**:

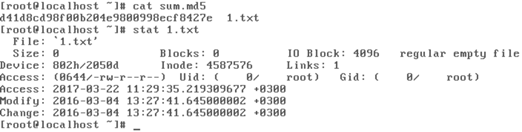
**$ touch -m -t 201601081830.14 файл**

Или дату последнего доступа:

**$ touch -a -t 201601081830.14 файл**

Чтобы посмотреть действительно ли задается информация, которую вы указали используйте команду **stat**:

**$ stat файл**



9) Утилита **dd**

Это утилита для копирования данных из одного файла в другой. Иногда необходимо создать файл определенного размера в **Linux**, тогда можно просто создать его на основе **/dev/zero** или **/dev/random**, вот так:

**dd if=/dev/zero of=~/файл count=20M**

Параметр **if** указывает откуда брать данные, а **of** - куда записывать, **count** - необходимый размер. Еще можно указать размер блока для записи с помощью **bs**, чем больше размер блока тем быстрее будет выполняться копирование.

10 ) **Создание специальных файлов в Linux**

В Linux, кроме выше рассмотренных обычных текстовых и бинарных файлов существуют еще и специальные файлы. Это файлы сокетов и туннелей. Их нельзя создать обычными программами, но для этого существуют специальные утилиты смотрите подробнее в статье, ссылку на которую я дал вверху.

Это были основные (возможные) команды для создания файла в Linux.

=====================================================================

**6.**Создать **два файла объединить в один файл** и его содержимое просмотреть на экране:

Создать текстовые файлы: **file.1** и **file.2**

Конкатенация файлов (объединение) в один файл **file.12** :

**cat file.1 file.2 > file.12**

=====================================================================

Выполнить остальные пункты:

1. Вывести дату командой **date**. Продублировать вывод в файл отчёта **Фамилия\_Инициалы.txt**.
2. Последовательный переход в каталоги **/bin, /usr, /etc, /usr/bin**.

В каждом каталоге выполняется команда **ls** с различными ключами **(-a, -l, -F, -R).** Если вывод команды **ls** не умещается на экране, то можно воспользоваться командой **less** (напр. **ls -l | less**):

**ls -a** (вывод всех файлов каталога)

**ls -l** (информация о файле)

**ls -F** (папки каталога)

**ls -R** (выводит список файлов каталога и его подкаталогов)

Переход в каталог **/bin**:

**user@localhost:/bin$**

Выполнение команды ls с различными ключами. Дублировать вывод в файл отчёта **Фамилия\_Инициалы.txt**

1. Создать новую директорию **MyWork** в каталоге **/tmp/ikpiXX/pz04** и переместить в нее полученные файлы. **file.1, file.2** и
2. Вывести полную информацию обо всех файлах и проанализировать уровни доступа

(**ls -l**)

1. Добавить для всех трех файлов право выполнения членам группы и остальным пользователям (c помощью команды **chmod**)
2. Просмотреть атрибуты файлов (c помощью команды **ls -l file.1 -** чтение атрибутов файла)
3. Вывод в файл **Фамилия\_Инициалы.txt** значений переменных окружения **PATH, LANG, HOME**:

**echo PATH=$PATH>> Фамилия\_Инициалы.txt**

**echo LANG=$LANG>> Фамилия\_Инициалы.txt**

**echo HOME=$HOME>> Фамилия\_Инициалы.txt**

Просмотр полученного файла при помощи одной из команд cat / less / more:

**less Фамилия\_Инициалы.txt**

1. Создать еще один каталог (**mkdir MyWork2**)
2. Установить дополнительную связь объединенного файла с новым каталогом, но под другим именем:

**ln** **/tmp/ikpiXX/pz04/MyWork/file.12 /tmp/ikpiXX/pz04/MyWork2/file.34**

1. Создать символическую связь:

**ln** **–s /tmp/ikpiXX/pz04/MyWork/file.12 /tmp/ikpiXX/pz04/MyWork2/file.67**

1. Сделать текущим новый каталог и вывести на экран расширенный список информации о его файлах:

**cd MyWork2**

**ls -l**

1. Произвести поиск заданной последовательности символов в файлах текущей директории и получить перечень соответствующих файлов:

**grep –i 1 file.34** (**1**23)

1. Получить информацию об активных процессах и имена других пользователей:

**ps -l**

1. Добавляем в конец файла Фамилия\_Инициалы.txt две строки. Первая содержит текст: “Практическое занятие №04”. Вторая – имя и фамилию студента, выполнившего работу.

**echo " Практическое занятие №9" >> Фамилия\_Инициалы.txt**

**echo "ФИО" >> Фамилия Имя Отчество, группа ИКПИ№№**

1. Защитить преподавателю отчет о работе и удалить свои файлы и каталоги (созданные на занятии) по команде преподавателя после защиты.
2. Выйти из системы.