Команды в командной строке.

mkdir

* Команда MD ( MKDIR ) используется для создания каталогов в командной строке Windows.
* Формат командной строки:

MKDIR [диск:]путь

MD [диск:]путь

* mkdir -p – ключ указывает путь от текущего расположения, по которому следует создать каталог.

.. – подняться выше

. – для самой директории

* mkdir -m *mode* – указывает режим, который будет использоваться для новых директории

Можно создавать несколько каталогов подряд (mkdir man woman hello world)

echo

* Применяется для вывода текстовых сообщений на стандартный вывод и для переключения режима отображения команд на экране.
* Формат командной строки:

ECHO [ON | OFF] - включить / выключить режим отображения вводимых команд на экране.

ECHO [сообщение] отобразить текст сообщения на экране.

* echo. – вывод пустой строки
* ECHO - при вводе команды без параметров, отображается текущий режим отображения команд
* @echo off - отключить режим вывода команд на экран. Символ @ перед командой echo используется для того, чтобы на экран не выводился и текст самой команды.
* echo -n – строка (\a – alert символ, \b – backspace, \c – печатать строку без новой строки, все символы после \c игнорируются, \f – форма подачи, \n – новая линия, \r – возврат каретки, \t – таб, \v – вертикальный таб, \\ - backslash, \0n – где n – код аски 8 бит (1-3 цифры)) – Хрен пойми, почему n не работает, но все остальное – только если записывается в двойные кавычки

cat

* Утилита cat в Linux читает последовательно файлы и выдает их содержимое в стандартный выходной поток.
* Эквивалентом cat в Windows является команда type, которая работает как из командной строки (CMD), так и в Windows PowerShell.
* Соединяет и выводит файлы
* cat file.txt – вывести на экран содержимое
* echo “line from first” > file.txt - создать файлы
* cat file.txt file2.txt > result.txt – объединяет файл
* cat result.txt – выводит результат как из первого, так и из второго

> - создать пустой файлик твоего формата

man – искать, что делает нужная команда

touch

* Изменяет временные метки файлов; если не создан файл, то он его создает (изменяет все)
* Чтобы изменить только время доступа к файлу, следует использовать ключ -a
* Чтобы изменить только время изменения файла, следует использовать ключ -m
* Чтобы использовать временные метки с другого файла, нужно использовать ключ -r <source> <target>
* Вы можете использовать опцию -t и временную метку в следующем формате:

[[CC]YY]MMDDhhmm[.ss]

CC – первые две цифры года

YY – Две две цифры года

MM – месяц года (01-12)

DD – День месяца (01-31)

hh – час дня (00-23)

mm – минута часа (00-59)

ss – Секунды (00-59)

* В приведенном выше случае CC является необязательным. По факту, CCYY также является необязательным, в этом случае он будет занимать текущий год. Аналогично, секунды также являются необязательными, по умолчанию они равны 00.
* Вы также можете использовать команду touch с символическими ссылками. Вам просто нужно использовать опцию -h, имея дело с символическими ссылками. Остальные остаются такими же, как и обычные файлы.
* -с – не создает файл, если его не существует
* -d – использует специфическое время, вместо текущего ( YYYY-MM-DDThh:mm:SS[.frac][tz] )

ls

* Список контента в директории (можно указывать расположение файла)
* -1 – выводит в одну колонку
* -С – дефолтный вывод
* -a – выводит в список все, включая файлы, начинающиеся с точки \*
* -A – как предыдущее, но без рабочей директории (.)и родительской директории(..)
* -b -заставляет печатать не принтабельные символы
* -c Uses time of last modification of the i-node (file created, mode changed, and so forth) for sorting (-t) or printing (-l or -n).
* -d -если аргумент в директории, ты выводит только его имя(обычно используется с -l, чтобы получить статус директории)
* -e -тоже самое, что и -l, кроме вывода времени во втором и с одним форматом, кроме возраста
* -E – обратный формат даты (сначала год)
* f – каждый элемент интерпретируется как директория и список в слове находится в каждом слове (выключает l, t, s, r, включает a)
* g – тоже самое, что и -l, но не пишется владелец
* -h -все размеры скейлятся под человеческий формат (гигабуты, мегабуты и т.п другие буты)
* -H –
* -i – для каждого файлы выводит i-node number в первой колонке вывода
* -l – полный формат вывода
* -L -как и с H работает
* -m – файлы выводятся через запятую
* -n – тоже самое, что и l, но файлы UID и GID
* -o -группа не печатается (как l)
* -p -ставит / после каждого файла в директории
* -q -непечатающиеся символы печатаются в именах файлов как вопрос??
* -r – печатает в обратном порядке по дате
* -R – рекурсивно выводит субдиректории
* -s – показывает общее число файловых системных блоков потребленных для каждого файла выводимого
* -t – сортирует по временным рамкам (вместо по имени)
* -u – использует время последнего доступа вместо сортировки по имени
* -v – тоже самое, что и -l и там \*
* -V –
* -x – мультиколончатый вывод, сортируемый через строку больше, чем в столбец
* -@ -
* просто r – файл читаемый
* просто w – файл записываемый
* просто x – файл исполняемый
* просто - - доступ не предоставляется

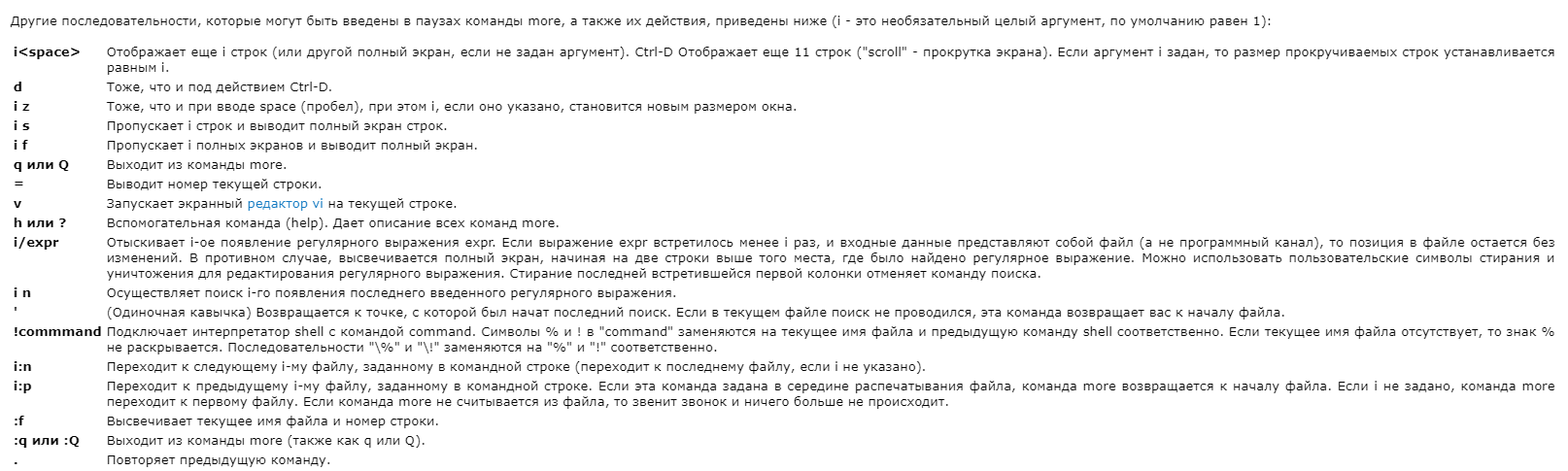
pwd

* Вывести полный путь до текущей рабочей директории, в которой находится пользователь.

cd

* -P – возвращает к корневой папке (а может просто не хочет читать ключ)
* / - возвращает в самое-самое начало (если после возврата написать cd – вернешься к себе домой)

more

* Простейшая программа для постраничного просмотра текста more используется с именем файла(ов) в качестве параметра или как последняя команда в цепочке перенаправлений ввода/вывода. С помощью опций +<число> или +/<образец текста> можно начать просмотр с интересующей строки. (полноэкранный просмотр текста файла)
* -с – Команда more отображает в верхней части экрана каждую страницу с начала, стирая при этом предварительно выведенную там строку. Это исключает прокрутку экрана, облегчая чтение текста с помощью команды more. Эта опция игнорируется, если терминал не имеет возможности очищать строку до конца.
* -d – Команда more в конце каждого заполненного экрана высвечивает сообщение "Hit space to continue, Rubout to abort" ("Нажмите пробел для продолжения, а Rubout - для прерывания"). Эта опция полезна, если команда more используется в качестве фильтра в некоторых системах, таких как класс, где многие пользователи могут быть неопытными.
* -s – сжать. Заменяет несколько пустых строк одной пустой строкой. Это полезно при выводе
* -f - Под действием этой опции команда more считает не экранные, а логические строки. То есть, длинные строки не заворачиваются. Рекомендуется использовать эту опцию, если вывод команды nroff подключается в конвейер через команду ul, которая может генерировать escape последовательности. Эти escape-последовательности содержат символы, которые обычно занимают экранные позиции, но не печатаются при выводе на терминал в качестве части escape-последовательности. Таким образом, команда more может решить, что строки длиннее, чем они есть на самом деле и ошибочно завернуть их.
* -l – Умышленно не обрабатывается команда Ctrl-L (form feed - переход на новую страницу). Если не задана эта опция, команда more останавливается после любой строки, содержащей Ctrl-L, до тех пор, пока экран не заполнится до конца. Также, если некоторый файл начинается со знака form feed, экран очищается прежде, чем распечатывается этот файл.
* -u - Обычно команда more осуществляет такое же подчеркивание, как при команде nroff, в том виде, который соответствует данному терминалу: если терминал может делать подчеркивание или имеет режим выделения, то выходные данные команды more соответствуют escape-последовательностям для разрешения подчеркивания или режима выделения для подчеркнутого текста в исходном файле. Опция -u подавляет такую обработку.
* -r - Обычно, команда more игнорирует управляющие символы, которые она не интерпретирует некоторым образом. Под действием опции -r эти символы отображаются как ^C, где стоит "С" для любого такого символа.
* -w - Обычно, команда more завершает работу при подходе к концу своих входных данных. А под действием опции -w, она запрашивает пользователя и ждет нажатия любой клавиши перед завершением работы.
* -lines – выводит кол-во линий на каждом полном экране больше, чем при дефолте
* +linenumber – начать со строки linenumber
* +/pattern - Команда more начинает просмотр текста за две строки до той строки, в которой содержится регулярное выражение pattern (шаблон)
* more [-cdflrsuw] [-n] [+linenumber] [+/pattern] [name...]

cp

* Утилита позволяет полностью копировать файлы и директории.
* -f – unlink. перезапись существующих файлов
* -H –
* -i – спрашивать, нужно ли перезаписывать файлы
* -L - копировать не символические ссылки, а то, на что они указывают
* -P - не следовать символическим ссылкам
* -r – копировать папку рекурсивно
* -R – тоже самое
* -p- сохранять владельца, временные метки и флаги доступа при копировании;

rm

* Удаление файлов и директорий
* rm опции файл(ы)
* r (R) - Удаление директорий и их содержимого. Рекурсивное удаление.
* -f -Игнорировать несуществующие файлы и аргументы. Никогда не выдавать запросы на подтверждение удаления.
* I - Выводить запрос на подтверждение удаления каждого файла.
* -p- разрешает юзерам удалять имя директории (dirname – пустая директория) и ее родительские директория, которые стали пустыми. Если одна из частей не может быть удалена, выводится ошибка
* -s- скрывает сообщение-ошибку от предыдущего ключа, в случае неудачи

rmdir (=rm -r)

* Ответ в том, что rmdir работает только в случае пустых каталогов - нет никакого способа, каким вы могли бы заставить rmdir удалить непустые каталоги.

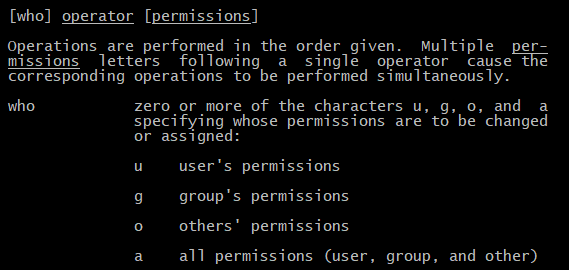
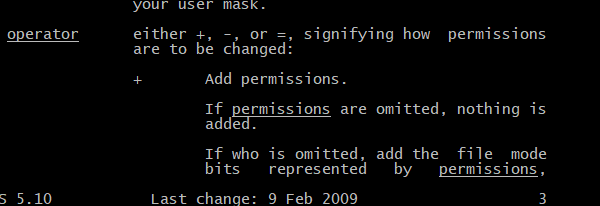
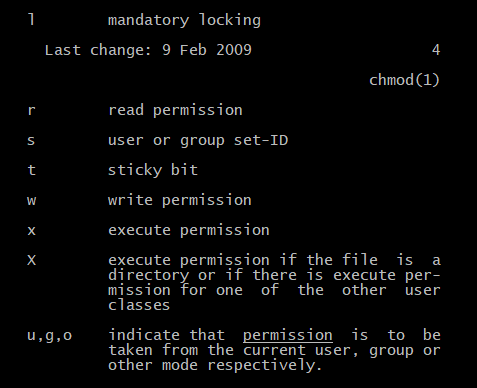
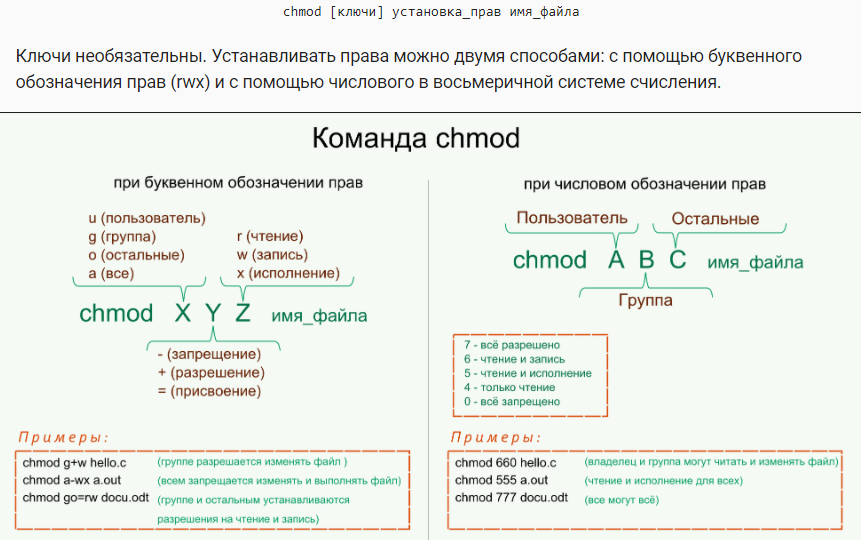
mv

* Перемещает файл в указанную директорию
* -f – переместит файлы без запроса, даже если он пишет поверх существующей цели. Если стандартный ввод не терминал, то это значение стоит по умолчанию
* -i – будет заказывать подтверждение при каждом движении

file \* - посмотреть все файлы в текущей папке (можно вместо \* указать имя)

find \*

chmod

* 
* 
* 
* A- removes all ACEs for current ACL on file and replaces current ACL with new ACL that represents only the current mode of the file.
* 
* Кроме того, часто требуется рекурсивно назначить права для вложенных в каталог объектов. Для этого используется ключ -R.
* В первой группе указывается, кому будет предоставляться или запрещаться доступ: владельцу (u), группе (g), остальным (o) или всем (a). Можно указать как одну, так сразу несколько категорий.
* Вторая группа обозначает запрещение, разрешение или назначение права и всегда состоит из одного символа: -, + или =.
* В третьей группе перечисляется изменяемые права: чтение (r), запись (w), исполнение (x). Можно указывать сразу несколько прав.

umask

ln (link)

* ln [-Ffhinsv] исходный\_файл [ целевой\_файл ]
* ln [-Ffhinsv] исходный\_файл ... целевой\_каталог
* link исходный\_файл целевой\_файл
* Делает жесткие или символьные ссылки на файл
* -f - Связывает (link) файлы, не спрашивая пользователя, даже если режим цели запрещает запись. Это по умолчанию, если стандартный ввод не является терминалом.
* -s – Создать символьную ссылку. В качестве первого аргумента пишется абсолютный адрес файла или каталога, на который делается ссылка, в качестве второго – абсолютный или относительный адрес создаваемой мягкой ссылки.
* -n – Если цель – существующий файл, пишет диагностическое письмо для stderr и переходит к оставшимся исходным файлам. Параметр -f отменяет этот параметр

wc

* Выводит кол-во строк, слов, или символов в файле
* -с – считает байты
* -l – считает строки
* -m – считает символы
* -С – то же самое, что и -m
* -w - Считает слова, разделенные пробелами или символы новой строки.

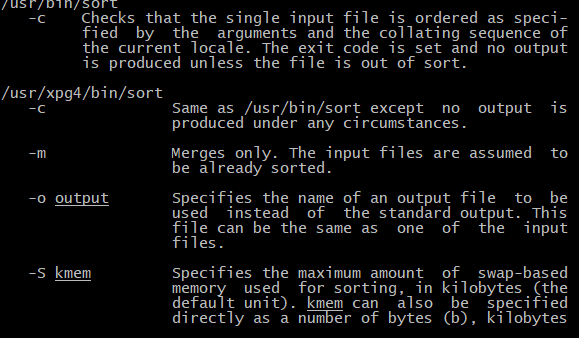
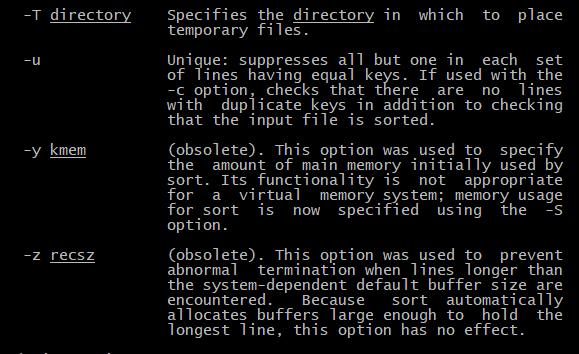
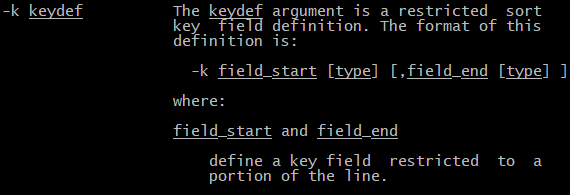
head

* Выводит первые несколько строк файлов
* -n number – первое число строк (целое положительное число), которое будет скопировано на стандартный вывод
* -number – целое положительное число, которое обозначает, сколько строк вывести (как предыдущее)
* head [options] [files]

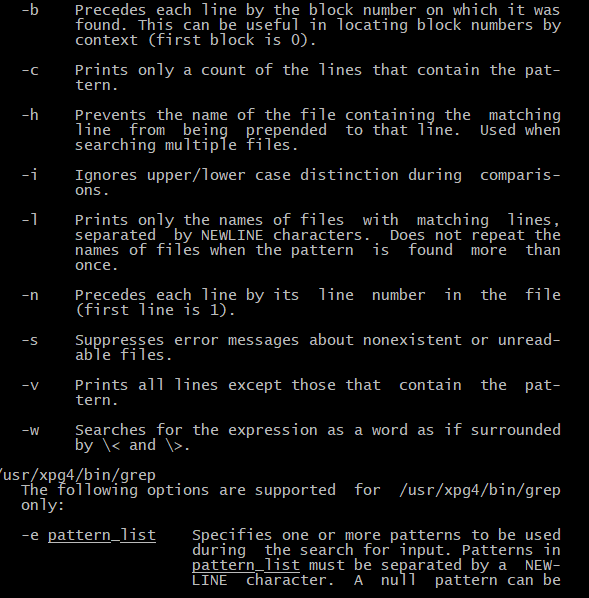
tail

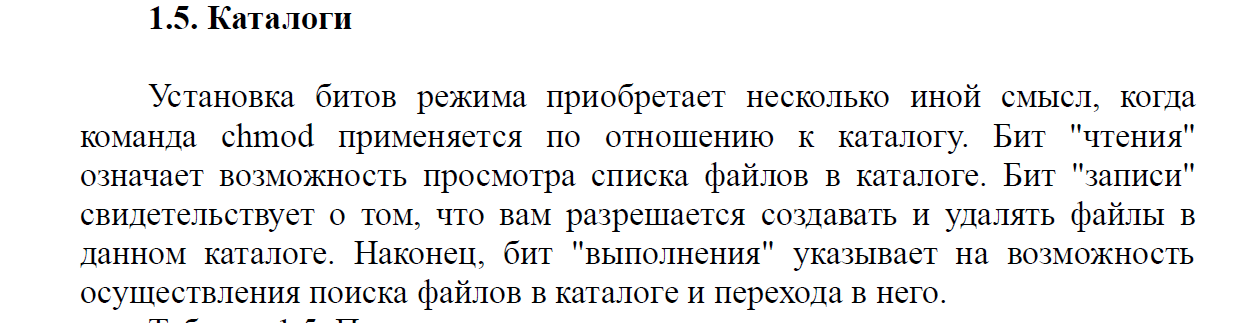
* $ tail опции файл
* -b – units of blocks( -- --)
* -c – units of bytes (выводить указанное количество байт с конца файла) (+ - копирование с начала файла, - - копирование с конца файла, = - с конца файла)
* -n number (= c number)
* -f - обновлять информацию по мере появления новых строк в файле
* -l – количество строк
* -r – обратный, копирует строчки с определенной точки в обратном порядке. =напечатать файл в обратном порядке, если просто default r)

sort

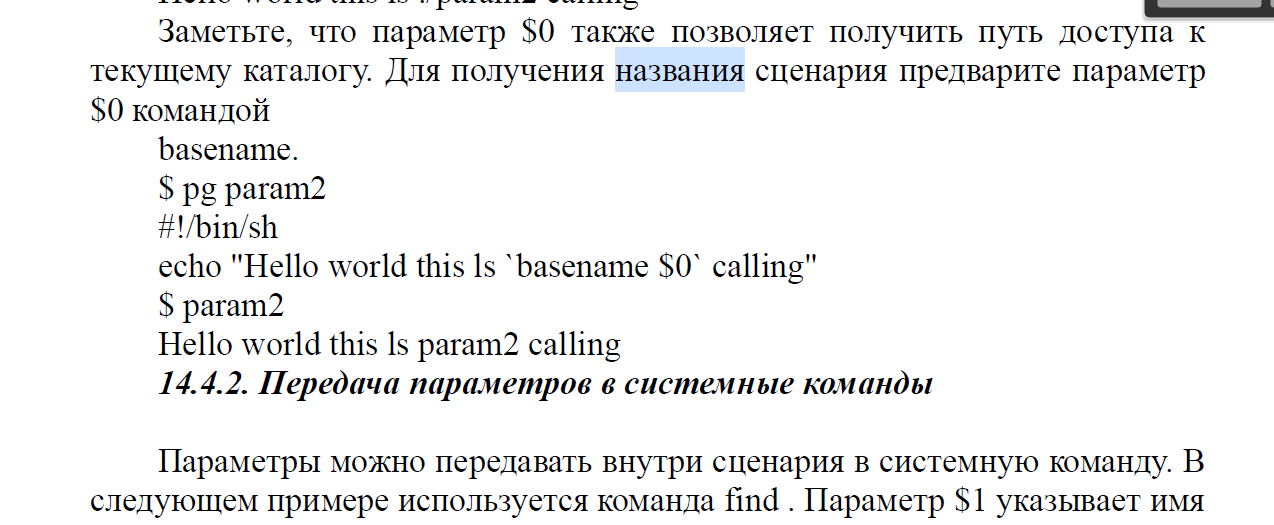
* сортировать, объединять или проверять последовательность текстовых файлов
* 
* 
* 
* -M, —month-sort сравнивать по месяцам в соответствии со следующим правилом [Любое неизвестное значение]< `JAN’ < … < `DEC’.
* -n, —numeric-sort сравнивать числовые значения строк без преобразования в числовой формат
* -r, —reverse вывод результатов в обратном порядке -o ФАЙЛ вывод результатов в указанный файл -k ПОЛЕ1, [ПОЛЕ2, ПОЛЕ3…] – сортировка по полям в заданном порядке
* -u, —unique – удалять дублирующие записи в результатах сортировки
* -c – проверка сортировки
* -b, —ignore-leading-blanks игнорировать пробелы в начале
* -t char– использовать символ char в качестве разделителя полей
* Команда sort не изменяет исходный файл, а просто выводит его содержимое в отсортированном виде. Чтобы сохранить результаты сортировки, воспользуйтесь опцией -o или перенаправлением вывода:
* 

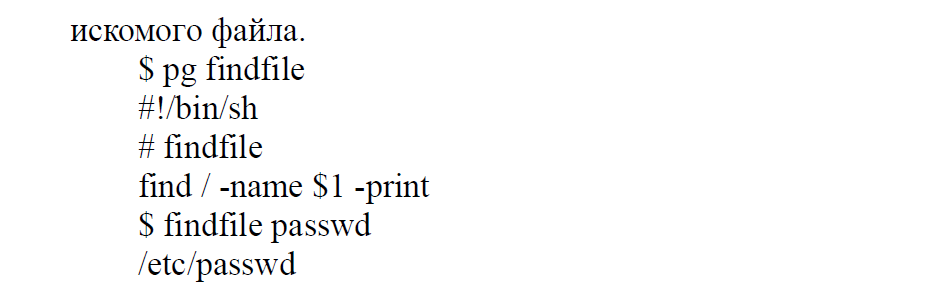
grep

* ищет в файле шаблон
* 
* prepend – добавлять
* precedes - предшествовать



Если в конце названия общего ресурса проставить знак доллара - $, то папка будет не видна из сети, это так называемый скрытый разделяемый ресурс и обратиться к ней можно только указав полный путь, например: "Пуск", "Выполнить", \\mycomp\sharing$, “OK”.





Жесткая ссылка работает как зеркальная копия оригинального файла. Эти ссылки используют одни и те же индексные дескрипторы (иноды). Изменения в одном из файлов, связанных жесткой ссылкой, отразятся в другом. При удалении жесткой ссылки с другим файлом ничего не произойдет. Жесткие ссылки не могут связывать различные файловые системы.

Так в чём собственно прикол? Прикол в том, что как уже упоминалось выше, жёсткая ссылка ссылается на индекс файла (inode). Поэтому, например, если мы удалим любой из файлов, а затем создадим файл в этом же месте файл с таким же именем (и расширением), то кажется, что он должен быть синхронизирован с другим файлом, но не тут то было — связь потеряна: так как новый файл — новый индекс (то же самое и при копировании файла — создаётся новый файл с новым индексом и связь теряется).

Во-первых, это ограничения. Жесткую ссылку нельзя создать на другом разделе - между разными файловыми системами могут работать только символьные ссылки. Жесткую ссылку нельзя создать на директорию - на директорию создаются только символьные ссылки.

При создании жёсткой ссылки создаётся новый путь (указатель) к тому же файлу. Можно удалить старый путь, а новый продолжит работать.

Создать каталог можно с помощью mkdir, mkdir с ключом p, можно создать родительский каталог, указав путь на дочерний

Файл можно создать с помощью >, touch, echo

$ ls dir | tee filename

По умолчанию команда tee создает новый файл с указанным именем, или перезаписывает одноименный, если он существовал ранее. Однако данная с опцией -a, она добавляет новые данные в конец существующего файла.

touch filename

Разница между рекурсией и псевдо-рекурсией (итерации) заключается в том, что итеративный процесс постоянно говорит «я сейчас посчитаю все что можно и продолжу» на каждом шаге рекурсии. Ему не нужно ничего запоминать вне вызова, он всегда считает все в первый возможный момент, и каждый шаг рекурсии может существовать в изоляции от прошлых, потому что вся информация передается из шага в шаг.

Ctrl+C: Прервать (убить) текущий процесс, запущенный в терминале. Данное сочетание клавиш посылает сигнал SIGINT процессу, который является стандартным запросом для большинства процессов, но некоторые могут проигнорировать его (скорее исключение, чем правило).

Ctrl+Z: Приостановить текущий процесс переднего плана, запущенный в bash терминале. Это посылает сигнал SIGTSTP процессу.

Ctrl+D: Закрывает оболочку bash. Это отправляет маркер EOF (конец файла) в bash, и bash завершает работу, когда он получает этот маркер.

Ctrl+L: Очистка экрана.

Ctrl+S: Остановить вывод на экран. Это особенно полезно при выполнении команд с большим количеством выходных данных, и вы не хотите останавливать саму команду с помощью Ctrl+C.

Ctrl+Q: Возобновите вывод на экран после его остановки с помощью Ctrl+S.

Ctrl+A или Home: перейти к началу строки.

Ctrl+E или End: перейти к концу строки.

Alt+B: перейти на одно слово влево (назад).

Ctrl+B: перейти на один символ влево (назад).

Alt+F: перейти на одно слово вправо (вперед) . – хз. Не знаю, работают ли эти штуки.

Ctrl+F: перейти на один символ вправо (вперед).

Ctrl+XX: перемещение между началом строки и текущей позицией курсора.

Ctrl+D или Delete: удалить символ до курсора.

Alt+D: удалить все символы после курсора на текущей строке.

Ctrl+H или Backspace: удалить символ перед курсором.

Строку, которая задана в качестве регулярного выражения и состоит из

нескольких слов, следует заключить в двойные кавычки. В противном

случае первое слово строки будет воспринято как образец поиска, а все

остальные слова будут считаться именами файлов. В итоге отобразится

сообщение о том, что указанные файлы не найдены.

Если образец поиска состоит из какой‑нибудь системной переменной,

например $PATH, рекомендуется тоже взять ее в двойные кавычки. Это

связано с тем, что, прежде чем передавать аргументы команде grep,

интерпретатор shell выполняет подстановку переменных, и команда grep

получает значение переменной, которое может содержать пробелы. В этом

случае будет выдано то же сообщение об ошибке, о котором говорилось в

предыдущем абзаце.

Шаблон поиска должен быть заключен в одинарные кавычки, если в

состав регулярного выражения входят метасимволы.

Если ничего в двойных кавычках нельзя интерполировать, то они будут работать как одинарные кавычки

r

Право чтения данного файла

w

Право записи/изменения данного файла

x

Право выполнения данного файла, если он является сценарием или

программой

(Для каталогов)

Возможность просмотра списка файлов в каталоге

Возможность создания/удаления файлов в каталоге

Возможность поиска файлов в каталоге и перехода в него

Биты смены идентификаторов (SUID и SGID)

1.6.1. Для чего нужны биты SUID и SGID?

1.6.2. Установка битов SUID и SGID

Биты SUID (Set User ID — установить идентификатор пользователя) и

SGID (Set Group ID — установить идентификатор группы)

Я создал несколько сценариев и установил для них бит

SG1D, благодаря чему пользователи, которым разрешен запуск этих

сценари

При запуске указанных сценариев пользователи

получают разрешение на выполнение операций по выгрузке базы данных,

хотя обычно такое право предоставляется только административному

персоналу. не обязаны регистрироваться а качестве администраторов баз

данных.

Чтобы установить бит SUID, вставьте цифру 4 перед числом, задающим

режим доступа. Биту SGID соответствует цифра 2. Если одновременно

устанавливаются оба бита, следует ввести цифру 6 (4 + 2).

В нашем примере видно, что команда направляет свой стандартный вывод в /dev/null (псевдоустройство, которое может принять произвольный объём данных, не сохраняя их совершенно нигде, следовательно, подавив стандартный вывод)

[command] & — запускает команду в фоне, т.е. управление отдаётся командному интерпретатору (bash, например), а [command] будет выполняться «параллельно».

[command] && — подразумевает, что следующая команда будет выполнена только в том случае, если [command] была выполнена успешна (вернула 0).

Мне необходимо для файлов поставить по умолчанию 777 при создании. Как это сделать?

если коротко, то — никак. длинное изложение — ниже, а в конце упомянуты соображения по поводу того, «что, если очень хочется…».

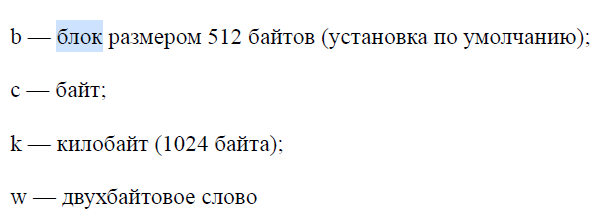
в конечном (для процесса) итоге файл/каталог создаётся функцией open() (см. $ man 1 open и упоминание в posix).

при этом во втором аргументе должен быть передан (в том числе и) бит O\_CREAT (для каталога — ещё и O\_DIRECTORY), а третьим аргументом должен идти набор бит, задающих разрешения процессам (в зависимости от того, от чьего имени они действуют) на доступ к создаваемому файлу/каталогу.

но указанный набор бит — не «окончательный приговор». как минимум на него может оказать влияение file mode creation mask (umask) — свойство процесса (наследуемое дочерними процессами), указывающее, какие биты разрешений надо скрыть («замаскировать») у создаваемого файла/каталога. именно «скрыть», но не добавить.

Произвести поиск файлов, размер которых превышает 10 блоков (5120

байтов), позволяет приведенная ниже команда:

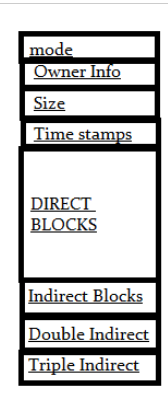


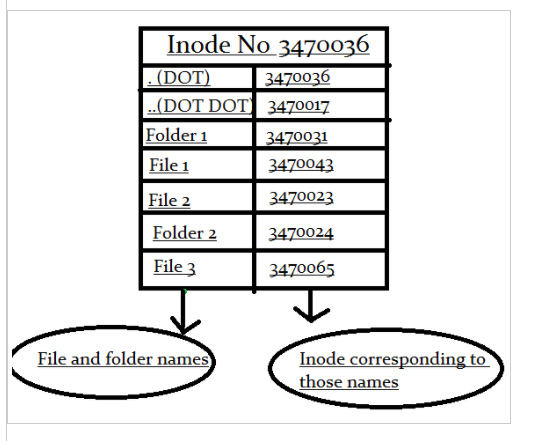
Проще говоря, в типичной простой файловой системе пространство делится на блоки . Типичный размер блока составляет 4 КБ. Каждый файл занимает целое число блоков. Если размер файла не кратен размеру блока, последний блок используется только частично. Таким образом, 1-байтовый файл и 4096-байтовый файл занимают 1 блок, тогда как 4097-байтовый файл занимает два блока. Вы можете наблюдать это с помощью du команды: если ваша файловая система имеет размер блока 4 КБ, то du сообщит о 4 КБ для 1-байтового файла.

512-байтовых единицах (секторах)

for и while в unix реализованы как и везде

Имена полезны для людей, но файловая система распознает файл не по имени, а по номеру. Номер, через который операционная система достигает местоположения и других атрибутов этого файла, называется номером inode.

<https://www.slashroot.in/inode-and-its-structure-linux>



Mode:

This keeps information about two things, one is the permission information, the other is the type of inode, for example an inode can be of a file, directory or a block device etc.

Owner Info: Access details like owner of the file, group of the file etc.

Size: This location store the size of the file in terms of bytes.

Time Stamps: it stores the inode creation time, modification time, etc.

Now comes the important thing to understand about how a file is saved in a partition with the help of an inode.

Block Size: Whenever a partition is formatted with a file system.It normally gets formatted with a default block size. Now block size is the size of chunks in which data

will be spread. So if the block size is 4K, then for a file of 15K it will take 4 blocks(because 4K\*4 16), and technically speaking you waste 1 K.

So after the 12 direct block pointers, 13th block pointer in inode is for Indirect block pointers, and 14th block pointer is for double indirect block pointers, and 15th block pointer is for triple indirect block pointers.