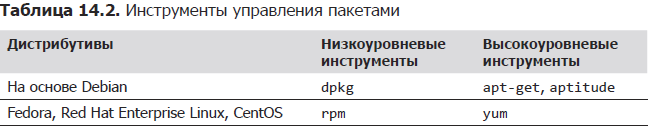
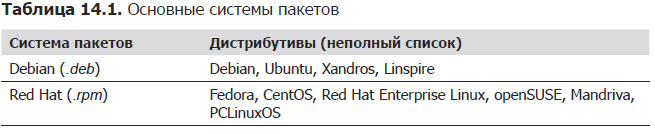
****

**Очень полезные команды Linux на одном листе**

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| **Системная информация** | |
| arch | отобразить архитектуру компьютера |
| uname -m |
| uname -r | отобразить используемую версию ядра |
| dmidecode -q | показать аппаратные системные компоненты - (SMBIOS / DMI) |
| hdparm -i /dev/hda | вывести характеристики жесткого диска |
| hdparm -tT /dev/sda | протестировать производительность чтения данных с жесткого диска |
| cat /proc/cpuinfo | отобразить информацию о процессоре |
| cat /proc/interrupts | показать прерывания |
| cat /proc/meminfo | проверить использование памяти |
| cat /proc/swaps | показать файл(ы) подкачки |
| cat /proc/version | вывести версию ядра |
| cat /proc/net/dev | показать сетевые интерфейсы и статистику по ним |
| cat /proc/mounts | отобразить смонтированные файловые системы |
| lspci -tv | показать в виде дерева PCI устройства |
| lsusb -tv | показать в виде дерева USB устройства |
| date | вывести системную дату |
| cal 2007 | вывести таблицу-календарь 2007-го года |
| date 041217002007.00 | установить системные дату и время ММДДЧЧммГГГГ.СС (МесяцДеньЧасМинутыГод.Секунды) |
| clock -w | сохранить системное время в BIOS |
| **Остановка системы** | |
| shutdown -h now | Остановить систему |
| init 0 |
| telinit 0 |
| shutdown -h hours:minutes & | запланировать остановку системы на указанное время |
| shutdown -c | отменить запланированную по расписанию остановку системы |
| shutdown -r now | перегрузить систему |
| reboot |
| logout | выйти из системы |
| **Файлы и директории** | |
| cd /home | перейти в директорию '/home' |
| cd .. | перейти в директорию уровнем выше |
| cd ../.. | перейти в директорию двумя уровнями выше |
| cd | перейти в домашнюю директорию |
| cd ~user | перейти в домашнюю директорию пользователя user |
| cd - | перейти в директорию, в которой находились до перехода в текущую директорию |
| pwd | показать текущую директорию |
| ls | отобразить содержимое текущей директории |
| ls -F | отобразить содержимое текущей директории с добавлением к именам символов, храктеризующих тип |
| ls -l | показать детализированое представление файлов и директорий в текущей директории |
| ls -a | показать скрытые файлы и директории в текущей директории |
| ls \*[0-9]\* | показать файлы и директории содержащие в имени цифры |
| tree | показать дерево файлов и директорий, начиная от корня (/) |
| lstree |
| mkdir dir1 | создать директорию с именем 'dir1' |
| mkdir dir1 dir2 | создать две директории одновременно |
| mkdir -p /tmp/dir1/dir2 | создать дерево директорий |
| rm -f file1 | удалить файл с именем 'file1' |
| rmdir dir1 | удалить директорию с именем 'dir1' |
| rm -rf dir1 | удалить директорию с именем 'dir1' и рекурсивно всё её содержимое |
| rm -rf dir1 dir2 | удалить две директории и рекурсивно их содержимое |
| mv dir1 new\_dir | переименовать или переместить файл или директорию |
| cp file1 file2 | сопировать файл file1 в файл file2 |
| cp dir/\* . | копировать все файлы директории dir в текущую директорию |
| cp -a /tmp/dir1 . | копировать директорию dir1 со всем содержимым в текущую директорию |
| cp -a dir1 dir2 | копировать директорию dir1 в директорию dir2 |
| ln -s file1 lnk1 | создать символическую ссылку на файл или директорию |
| ln file1 lnk1 | создать "жёсткую" (физическую) ссылку на файл или директорию |
| touch -t 0712250000 fileditest | модифицировать дату и время создания файла, при его отсутствии, создать файл с указанными датой и временем (YYMMDDhhmm) |
| **Поиск файлов** | |
| find / -name file1 | найти файлы и директории с именем file1. Поиск начать с корня (/) |
| find / -user user1 | найти файл и директорию принадлежащие пользователю user1. Поиск начать с корня (/) |
| find /home/user1 -name "\*.bin" | Найти все файлы и директории, имена которых оканчиваются на '. bin'. Поиск начать с '/ home/user1' |
| find /usr/bin -type f -atime +100 | найти все файлы в '/usr/bin', время последнего обращения к которым более 100 дней |
| find /usr/bin -type f -mtime -10 | найти все файлы в '/usr/bin', созданные или изменённые в течении последних 10 дней |
| find / -name \*.rpm -exec chmod 755 '{}' \; | найти все фалы и директории, имена которых оканчиваются на '.rpm', и изменить права доступа к ним |
| find / -xdev -name "\*.rpm" | найти все фалы и директории, имена которых оканчиваются на '.rpm', игнорируя съёмные носители, такие как cdrom, floppy и т.п. |
| locate "\*.ps" | найти все файлы, сожержащие в имени '.ps'. Предварительно рекомендуется выполнить команду 'updatedb' |
| whereis halt | показывает размещение бинарных файлов, исходных кодов и руководств, относящихся к файлу 'halt' |
| which halt | отображает полный путь к файлу 'halt' |
| **Монтирование файловых систем** | |
| mount /dev/hda2 /mnt/hda2 | монтирует раздел 'hda2' в точку монтирования '/mnt/hda2'. Убедитесь в наличии директории-точки монтирования '/mnt/hda2' |
| umount /dev/hda2 | размонтирует раздел 'hda2'. Перед выполнением, покиньте '/mnt/hda2' |
| fuser -km /mnt/hda2 | принудительное размонтирование раздела. Применяется в случае, когда раздел занят каким-либо пользователем |
| umount -n /mnt/hda2 | выполнить размонитрование без занесения информации в /etc/mtab. Полезно когда файл имеет атрибуты "только чтение" или недостаточно места на диске |
| mount /dev/fd0 /mnt/floppy | монтировать флоппи-диск |
| mount /dev/cdrom /mnt/cdrom | монтировать CD или DVD |
| mount /dev/hdc /mnt/cdrecorder | монтировать CD-R/CD-RW или DVD-R/DVD-RW(+-) |
| mount -o loop file.iso /mnt/cdrom | смонтировать ISO-образ |
| mount -t vfat /dev/hda5 /mnt/hda5 | монтировать файловую систему Windows FAT32 |
| mount -t smbfs -o username=user,password=pass //winclient/share /mnt/share | монтировать сетевую файловую систему Windows (SMB/CIFS) |
| mount -o bind /home/user/prg /var/ftp/user | "монтирует" директорию в директорию (binding). Доступна с версии ядра 2.4.0. Полезна, например, для предоставления содержимого пользовательской директории через ftp при работе ftp-сервера в "песочнице" (chroot), когда симлинки сделать невозможно. |
| **Дисковое пространство** | |
| df -h | отображает информацию о смонтированных разделах с отображением общего, доступного и используемого пространства *(Прим.переводчика. ключ -h работает не во всех \*nix системах)* |
| ls -lSr |more | выдаёт список файлов и директорий рекурсивно с сортировкой по возрастанию размера и позволяет осуществлять постраничный просмотр |
| du -sh dir1 | подсчитывает и выводит размер, занимаемый директорией 'dir1' *(Прим.переводчика. ключ -h работает не во всех \*nix системах)* |
| du -sk \* | sort -rn | отображает размер и имена файлов и директорий, с соритровкой по размеру |
| rpm -q -a --qf '%10{SIZE}\t%{NAME}\n' | sort -k1,1n | показывает размер используемого дискового пространства, занимаемое файлами rpm-пакета, с сортировкой по размеру (fedora, redhat и т.п.) |
| dpkg-query -W -f='${Installed-Size;10}\t${Package}\n' | sort -k1,1n | показывает размер используемого дискового пространства, занимаемое файлами deb-пакета, с сортировкой по размеру (ubuntu, debian т.п.) |
| **Пользователи и группы** | |
| groupadd group\_name | создать новую группу с именем group\_name |
| groupdel group\_name | удалить группу group\_name |
| groupmod -n new\_group\_name old\_group\_name | переименовать группу old\_group\_name в new\_group\_name |
| useradd -c "Nome Cognome" -g admin -d /home/user1 -s /bin/bash user1 | создать пользователя user1, назначить ему в качестве домашнего каталога /home/user1, в качестве shell'а /bin/bash, включить его в группу admin и добавить комментарий Nome Cognome |
| useradd user1 | создать пользователя user1 |
| userdel -r user1 | удалить пользователя user1 и его домашний каталог |
| usermod -c "User FTP" -g system -d /ftp/user1 -s /bin/nologin user1 | изменить атрибуты пользователя |
| passwd | сменить пароль |
| passwd user1 | сменить пароль пользователя user1 (только root) |
| chage -E 2005-12-31 user1 | установить дату окончания действия учётной записи пользователя user1 |
| pwck | проверить корректность системных файлов учётных записей. Проверяются файлы /etc/passwd и /etc/shadow |
| grpck | проверяет корректность системных файлов учётных записей. Проверяется файл/etc/group |
| newgrp [-] group\_name | изменяет первичную группу текущего пользователя. Если указать "-", ситуация будет идентичной той, в которой пользователь вышил из системы и снова вошёл. Если не указывать группу, первичная группа будет назначена из /etc/passwd |
| **Выставление/изменение полномочий на файлы** | |
| ls -lh | просмотр полномочий на файлы и директории в текущей директории |
| ls /tmp | pr -T5 -W$COLUMNS | вывести содержимое директории /tmp и разделить вывод на пять колонок |
| chmod ugo+rwx directory1 | добавить полномочия на директорию directory1 ugo(User Group Other)+rwx(Read Write eXecute) - всем полные права. Аналогичное можно сделать таким образом "chmod 777 directory1" |
| chmod go-rwx directory1 | отобрать у группы и всех остальных все полномочия на директорию directory1. |
| chown user1 file1 | назначить владельцем файла file1 пользователя user1 |
| chown -R user1 directory1 | назначить рекурсивно владельцем директории directory1 пользователя user1 |
| chgrp group1 file1 | сменить группу-владельца файла file1 на group1 |
| chown user1:group1 file1 | сменить владельца и группу владельца файла file1 |
| find / -perm -u+s | найти, начиная от корня, все файлы с выставленным SUID |
| chmod u+s /bin/binary\_file | назначить SUID-бит файлу /bin/binary\_file. Это даёт возможность любому пользователю запускать на выполнение файл с полномочиями владельца файла. |
| chmod u-s /bin/binary\_file | снять SUID-бит с файла /bin/binary\_file. |
| chmod g+s /home/public | назначить SGID-бит директории /home/public. |
| chmod g-s /home/public | снять SGID-бит с директории /home/public. |
| chmod o+t /home/public | назначить STIKY-бит директории /home/public. Позволяет удалять файлы только владельцам |
| chmod o-t /home/public | снять STIKY-бит с директории /home/public |
| **Специальные атрибуты файлов** | |
| chattr +a file1 | позволить открывать файл на запись только в режиме добавления |
| chattr +c file1 | позволяет ядру автоматически сжимать/разжимать содержимое файла. |
| chattr +d file1 | указавет утилите dump игнорировать данный файл во время выполнения backup'а |
| chattr +i file1 | делает файл недоступным для любых изменений: редактирование, удаление, перемещение, создание линков на него. |
| chattr +s file1 | позволяет сделать удаление файла безопасным, т.е. выставленный атрибут s говорит о том, что при удалении файла, место, занимаемое файлом на диске заполняется нулями, что предотвращяет возможность восстановления данных. |
| chattr +S file1 | указывает, что, при сохранении изменений, будет произведена синхронизация, как при выполнении команды sync |
| chattr +u file1 | данный атрибут указывает, что при удалении файла содержимое его будет сохранено и при необходимости пользователь сможет его восстановить |
| lsattr | показать атрибуты файлов |
| **Архивирование и сжатие файлов** | |
| bunzip2 file1.bz2 | разжимает файл 'file1.gz' |
| gunzip file1.gz |
| gzip file1 | сжимает файл 'file1' |
| bzip2 file1 |
| gzip -9 file1 | сжать файл file1 с максимальным сжатием |
| rar a file1.rar test\_file | создать rar-архив 'file1.rar' и включить в него файл test\_file |
| rar a file1.rar file1 file2 dir1 | создать rar-архив 'file1.rar' и включить в него file1, file2 и dir1 |
| unrar x file1.rar | распаковать rar-архив |
| tar -cvf archive.tar file1 | создать tar-архив archive.tar, содержащий файл file1 |
| tar -cvf archive.tar file1 file2 dir1 | создать tar-архив archive.tar, содержащий файл file1, file2 и dir1 |
| tar -tf archive.tar | показать содержимое архива |
| tar -xvf archive.tar | распаковать архив |
| tar -xvf archive.tar -C /tmp | распаковать архив в /tmp |
| tar -cvjf archive.tar.bz2 dir1 | создать архив и сжать его с помощью bzip2*(Прим.переводчика. ключ -j работает не во всех \*nix системах)* |
| tar -xvjf archive.tar.bz2 | разжать архив и распаковать его*(Прим.переводчика. ключ -j работает не во всех \*nix системах)* |
| tar -cvzf archive.tar.gz dir1 | создать архив и сжать его с помощью gzip |
| tar -xvzf archive.tar.gz | разжать архив и распаковать его |
| zip file1.zip file1 | создать сжатый zip-архив |
| zip -r file1.zip file1 file2 dir1 | создать сжатый zip-архив и со включением в него нескольких файлов и/или директорий |
| unzip file1.zip | разжать и распаковать zip-архив |
| **RPM пакеты (Fedora, Red Hat и тому подобное)** | |
| rpm -ivh package.rpm | установить пакет с выводом сообщений и прогресс-бара |
| rpm -ivh --nodeps package.rpm | установить пакет с выводом сообщений и прогресс-бара без контроля зависимостей |
| rpm -U package.rpm | обновить пакет без изменений конфигурационных файлов, в случае отсутствия пакета, он будет установлен |
| rpm -F package.rpm | обновить пакет только если он установлен |
| rpm -e package\_name.rpm | удалить пакет |
| rpm -qa | отобразить список всех пакетов, установленных в системе |
| rpm -qa | grep httpd | среди всех пакетов, установленных в системе, найти пакет содержащий в своём имени "httpd" |
| rpm -qi package\_name | вывести информацию о конкрентном пакете |
| rpm -qg "System Environment/Daemons" | отобразить пакеты входящие в группу пакетов |
| rpm -ql package\_name | вывести список файлов, входящих в пакет |
| rpm -qc package\_name | вывести список конфигурационных файлов, входящих в пакет |
| rpm -q package\_name --whatrequires | вывести список пакетов, необходимых для установки конкретного пакета по зависимостям |
| rpm -q package\_name --whatprovides | show capability provided by a rpm package |
| rpm -q package\_name --scripts | отобразит скрипты, запускаемые при установке/удалении пакета |
| rpm -q package\_name --changelog | вывести историю ревизий пакета |
| rpm -qf /etc/httpd/conf/httpd.conf | проверить какому пакету принадлежит указанный файл. Указывать следует полный путь и имя файла. |
| rpm -qp package.rpm -l | отображает список файлов, входящих в пакет, но ещё не установленных в систему |
| rpm --import /media/cdrom/RPM-GPG-KEY | ипортировать публичный ключ цифровой подписи |
| rpm --checksig package.rpm | проверит подпись пакета |
| rpm -qa gpg-pubkey | проверить целостность установленного содержимого пакета |
| rpm -V package\_name | проверить размер, полномочия, тип, владельца, группу, MD5-сумму и дату последнего изменеия пакета |
| rpm -Va | проверить содержимое всех пакеты установленные в систему. Выполняйте с осторожностью! |
| rpm -Vp package.rpm | проверить пакет, который ещё не установлен в систему |
| rpm2cpio package.rpm | cpio --extract --make-directories \*bin\* | извлечь из пакета файлы содержащие в своём имени bin |
| rpm -ivh /usr/src/redhat/RPMS/`arch`/package.rpm | установить пакет, собранный из исходных кодов |
| rpmbuild --rebuild package\_name.src.rpm | собрать пакет из исходных кодов |
| **YUM - средство обновления пакетов(Fedora, RedHat и тому подобное)** | |
| yum install package\_name | закачать и установать пакет |
| yum update | обновить все пакеты, установленные в систему |
| yum update package\_name | обновить пакет |
| yum remove package\_name | удалить пакет |
| yum list | вывести список всех пакетов, установленных в систему |
| yum search package\_name | найти пакет в репозитории |
| yum clean packages | очисть rpm-кэш, удалив закачанные пакеты |
| yum clean headers | удалить все заголовки файлов, которые система использует для разрешения зависимостей |
| yum clean all | очисть rpm-кэш, удалив закачанные пакеты и заголовки |
| **DEB пакеты (Debian, Ubuntu и тому подобное)** | |
| dpkg -i package.deb | установить / обновить пакет |
| dpkg -r package\_name | удалить пакет из системы |
| dpkg -l | показать все пакеты, установленные в систему |
| dpkg -l | grep httpd | среди всех пакетов, установленных в системе, найти пакет содержащий в своём имени "httpd" |
| dpkg -s package\_name | отобразить инфрмацию о конкретном пакете |
| dpkg -L package\_name | вывести список файлов, входящих в пакет, установленный в систему |
| dpkg --contents package.deb | отобразить список файлов, входящих в пакет, который ешё не установлен в систему |
| dpkg -S /bin/ping | найти пакет, в который входит указанный файл. |
| **APT - средство управление пакетами (Debian, Ubuntu и тому подобное)** | |
| apt-get install package\_name | установить / обновить пакет |
| apt-cdrom install package\_name | установить / обновить пакет с cdrom'а |
| apt-get update | получить обновлённые списки пакетов |
| apt-get upgrade | обновить пакеты, установленные в систему |
| apt-get remove package\_name | удалить пакет, установленный в систему с сохранением файлов конфигурации |
| apt-get purge package\_name | удалить пакет, установленный в систему с удалением файлов конфигурации |
| apt-get check | проверить целостность зависимостей |
| apt-get clean | удалить загруженные архивные файлы пакетов |
| apt-get autoclean | удалить старые загруженные архивные файлы пакетов |
| **Просмотр содержимого файлов** | |
| cat file1 | вывести содержимое файла file1 на стандартное устройсво вывода |
| tac file1 | вывести содержимое файла file1 на стандартное устройсво вывода в обратном порядке (последняя строка становиться первой и т.д.) |
| more file1 | постраничный вывод содержимого файла file1 на стандартное устройство вывода |
| less file1 | постраничный вывод содержимого файла file1 на стандартное устройство вывода, но с возможностью пролистывания в обе стороны (вверх-вниз), поиска по содержимому и т.п. |
| head -2 file1 | вывести первые две строки файла file1 на стандартное устройство вывода. По-умолчанию выводится десять строк |
| tail -2 file1 | вывести последние две строки файла file1 на стандартное устройство вывода. По-умолчанию выводится десять строк |
| tail -f /var/log/messages | выводить содержимое файла /var/log/messages на стандартное устройство вывода по мере появления в нём текста. |
| **Манипуляции с текстом** | |
| cat file | grep -i "Criteria" > result.txt | общий синтаксис выполнения действий по обработке содержимого файла и вывода результата в новый |
| cat file | grep -i "Criteria" >> result.txt | общий синтаксис выполнения действий по обработке содержимого файла и вывода результата в существующий файл. Если файл не существует, он будет создан |
| grep Aug /var/log/messages | из файла '/var/log/messages' отобрать и вывести на стандартное устройство вывода строки, содержащие "Aug" |
| grep ^Aug /var/log/messages | из файла '/var/log/messages' отобрать и вывести на стандартное устройство вывода строки, начинающиеся на "Aug" |
| grep [0-9] /var/log/messages | из файла '/var/log/messages' отобрать и вывести на стандартное устройство вывода строки, содержащие цифры |
| grep Aug -R /var/log/\* | отобрать и вывести на стандартное устройство вывода строки, содержащие "Aug", во всех файлах, находящихся в директории /var/log и ниже |
| sed 's/stringa1/stringa2/g' example.txt | в файле example.txt заменить "string1" на "string2", результат вывести на стандартное устройство вывода. |
| sed '/^$/d' example.txt | удалить пустые строки из файла example.txt |
| sed '/ \*#/d; /^$/d' example.txt | удалить пустые строки и комментарии из файла example.txt |
| echo 'test' | tr '[:lower:]' '[:upper:]' | преобразовать символы из нижнего регистра в верхний |
| sed -e '1d' result.txt | удалить первую строку из файла example.txt |
| sed -n '/string1/p' | отобразить только строки содержашие "string1" |
| sed -e 's/ \*$//' example.txt | удалить пустые символы в в конце каждой строки |
| sed -e 's/string1//g' example.txt | удалить строку "string1" из текста не изменяя всего остального |
| sed -n '1,8p;5q' example.txt | взять из файла с первой по восьмую строки и из них вывести первые пять |
| sed -n '5p;5q' example.txt | вывести пятую строку |
| sed -e 's/0\*/0/g' example.txt | заменить последовательность из любого количества нулей одним нулём |
| cat -n file1 | пронумеровать строки при выводе содержимого файла |
| cat example.txt | awk 'NR%2==1' | при выводе содержимого файла, не выводить чётные строки файла |
| echo a b c | awk '{print $1}' | вывести первую колонку. Разделение, по-умолчанию, по проблелу/пробелам или символу/символам табуляции |
| echo a b c | awk '{print $1,$3}' | вывести первую и треью колонки. Разделение, по-умолчанию, по проблелу/пробелам или символу/символам табуляции |
| paste file1 file2 | объединить содержимое file1 и file2 в виде таблицы: строка 1 из file1 = строка 1 колонка 1-n, строка 1 из file2 = строка 1 колонка n+1-m |
| paste -d '+' file1 file2 | объединить содержимое file1 и file2 в виде таблицы с разделителем "+" |
| sort file1 file2 | отсортировать содержимое двух файлов |
| sort file1 file2 | uniq | отсортировать содержимое двух файлов, не отображая повторов |
| sort file1 file2 | uniq -u | отсортировать содержимое двух файлов, отображая только уникальные строки (строки, встречающиеся в обоих файлах, не выводятся на стандартное устройство вывода) |
| sort file1 file2 | uniq -d | отсортировать содержимое двух файлов, отображая только повторяющиеся строки |
| comm -1 file1 file2 | сравнить содержимое двух файлов, не отображая строки принадлежащие файлу 'file1' |
| comm -2 file1 file2 | сравнить содержимое двух файлов, не отображая строки принадлежащие файлу 'file2' |
| comm -3 file1 file2 | сравнить содержимое двух файлов, удаляя строки встречающиеся в обоих файлах |
| **Преобразование наборов символов и файловых форматов** | |
| dos2unix filedos.txt fileunix.txt | конвертировать файл текстового формата из MSDOS в UNIX (разница в символах возврата коретки) |
| unix2dos fileunix.txt filedos.txt | конвертировать файл текстового формата из UNIX в MSDOS (разница в символах возврата коретки) |
| recode ..HTML < page.txt > page.html | конвертировать содержимое тестового файла page.txt в html-файл page.html |
| recode -l | more | вывести список доступных форматов |
| **Анализ файловых систем** | |
| badblocks -v /dev/hda1 | проверить раздел hda1 на наличие bad-блоков |
| fsck /dev/hda1 | проверить/восстановить целостность linux-файловой системы раздела hda1 |
| fsck.ext2 /dev/hda1 | проверить/восстановить целостность файловой системы ext2 раздела hda1 |
| e2fsck /dev/hda1 |  |
| e2fsck -j /dev/hda1 | проверить/восстановить целостность файловой системы ext3 раздела hda1 с указанием, что журнал расположен там же |
| fsck.ext3 /dev/hda1 | проверить/восстановить целостность файловой системы ext3 раздела hda1 |
| fsck.vfat /dev/hda1 | проверить/восстановить целостность файловой системы fat раздела hda1 |
| fsck.msdos /dev/hda1 |
| dosfsck /dev/hda1 |
| **Форматирование файловых систем** | |
| mkfs /dev/hda1 | создать linux-файловую систему на разделе hda1 |
| mke2fs /dev/hda1 | создать файловую систему ext2 на разделе hda1 |
| mke2fs -j /dev/hda1 | создать журналирующую файловую систему ext3 на разделе hda1 |
| mkfs -t vfat 32 -F /dev/hda1 | создать файловую систему FAT32 на разделе hda1 |
| fdformat -n /dev/fd0 | форматирование флоппи-диска без проверки |
| mkswap /dev/hda3 | создание swap-пространства на разделе hda3 |
| **swap-пространство** | |
| mkswap /dev/hda3 | создание swap-пространства на разделе hda3 |
| swapon /dev/hda3 | активировать swap-пространство, расположенное на разделе hda3 |
| swapon /dev/hda2 /dev/hdb3 | активировать swap-пространства, расположенные на разделах hda2 и hdb3 |
| **Создание резервных копий (backup)** | |
| dump -0aj -f /tmp/home0.bak /home | создать полную резервную копию директории /home в файл /tmp/home0.bak |
| dump -1aj -f /tmp/home0.bak /home | создать инкрементальную резервную копию директории /home в файл /tmp/home0.bak |
| restore -if /tmp/home0.bak | восстановить из резервной копии /tmp/home0.bak |
| rsync -rogpav --delete /home /tmp | синхронизировать /tmp с /home |
| rsync -rogpav -e ssh --delete /home ip\_address:/tmp | синхронизировать через SSH-туннель |
| rsync -az -e ssh --delete ip\_addr:/home/public /home/local | синхронизировать локальную директорию с удалённой директорией через ssh-туннель со сжатием |
| rsync -az -e ssh --delete /home/local ip\_addr:/home/public | синхронизировать удалённую директорию с локальной директорией через ssh-туннель со сжатием |
| dd bs=1M if=/dev/hda | gzip | ssh user@ip\_addr 'dd of=hda.gz' | сделать "слепок" локального диска в файл на удалённом компьютере через ssh-туннель |
| tar -Puf backup.tar /home/user | создать инкрементальную резервную копию директории '/home/user' в файл backup.tar с сохранением полномочий |
| ( cd /tmp/local/ && tar c . ) | ssh -C user@ip\_addr 'cd /home/share/ && tar x -p' | копирование содержимого /tmp/local на удалённый компьютер через ssh-туннель в /home/share/ |
| ( tar c /home ) | ssh -C user@ip\_addr 'cd /home/backup-home && tar x -p' | копирование содержимого /home на удалённый компьютер через ssh-туннель в /home/backup-home |
| tar cf - . | (cd /tmp/backup ; tar xf - ) | копирование одной директории в другую с сохранением полномочий и линков |
| find /home/user1 -name '\*.txt' | \  xargs cp -av --target-directory=/home/backup/ --parents | поиск в /home/user1 всех файлов, имена которых оканчиваются на '.txt', и копирование их в другую директорию |
| find /var/log -name '\*.log' | tar cv --files-from=- | bzip2 > log.tar.bz2 | поиск в /var/log всех файлов, имена которых оканчиваются на '.log', и создание bzip-архива из них |
| dd if=/dev/hda of=/dev/fd0 bs=512 count=1 | создать копию MBR (Master Boot Record) с /dev/hda на флоппи-диск |
| dd if=/dev/fd0 of=/dev/hda bs=512 count=1 | восстановить MBR с флоппи-диска на /dev/hda |
| **CDROM** | |
| cdrecord -v gracetime=2 dev=/dev/cdrom -eject blank=fast -force | clean a rewritable cdrom |
| mkisofs /dev/cdrom > cd.iso | create an iso image of cdrom on disk |
| mkisofs /dev/cdrom | gzip > cd\_iso.gz | create a compressed iso image of cdrom on disk |
| mkisofs -J -allow-leading-dots -R -V "Label CD" -iso-level 4 -o ./cd.iso data\_cd | create an iso image of a directory |
| cdrecord -v dev=/dev/cdrom cd.iso | burn an ISO image |
| gzip -dc cd\_iso.gz | cdrecord dev=/dev/cdrom - | burn a compressed ISO image |
| mount -o loop cd.iso /mnt/iso | mount an ISO image |
| cd-paranoia -B | rip audio tracks from a CD to wav files |
| cd-paranoia -- "-3" | rip first three audio tracks from a CD to wav files |
| cdrecord --scanbus | scan bus to identify the channel scsi |
| **Сеть (LAN и WiFi)** | |
| ifconfig eth0 | показать конфигурацию сетевого интерфейса eth0 |
| ifup eth0 | активировать (поднять) интерфейс eth0 |
| ifdown eth0 | деактивировать (опустить) интерфейс eth0 |
| ifconfig eth0 192.168.1.1 netmask 255.255.255.0 | выставить интерфейсу eth0 ip-адрес и маску подсети |
| ifconfig eth0 promisc | перевести интерфейс eth0 в promiscuous-режим для "отлова" пакетов (sniffing) |
| ifconfig eth0 -promisc | отключить promiscuous-режим на интерфейсе eth0 |
| dhclient eth0 | активировать интерфейс eth0 в dhcp-режиме. |
| route -n | вывести локальную таблицу маршрутизации |
| netstat -rn |
| route add -net 0/0 gw IP\_Gateway | задать ip-адрес шлюза по умолчанию (default gateway) |
| route add -net 192.168.0.0 netmask 255.255.0.0 gw 192.168.1.1 | добавить статический маршрут в сеть 192.168.0.0/16 через шлюз с ip-адресом 192.168.1.1 |
| route del 0/0 gw IP\_gateway | удалить ip-адрес шлюза по умолчанию (default gateway) |
| echo "1" > /proc/sys/net/ipv4/ip\_forward | разрешить пересылку пакетов (forwarding) |
| hostname | отобразить имя компьютера |
| host www.yandex.ru | разрешить имя www.yandex.ru хоста в ip-адрес и наоборот |
| host 93.158.134.3 |
| ip link show | отобразить состояние всех интерфейсов |
| mii-tool eth0 | отобразить статус и тип соединения для интерфейса eth0 |
| ethtool eth0 | отображает статистику интерфеса eth0 с выводом такой информации, как поддерживаемые и текущие режимы соединения |
| netstat -tupn | отображает все установленные сетевые соединения по протоколам TCP и UDP без разрешения имён в ip-адреса и PID'ы и имена процессов, обеспечивающих эти соединения |
| netstat -tupln | отображает все сетевые соединения по протоколам TCP и UDP без разрешения имён в ip-адреса и PID'ы и имена процессов, слушающих порты |
| tcpdump tcp port 80 | отобразить весь трафик на TCP-порт 80 (обычно - HTTP) |
| iwlist scan | просканировать эфир на предмет, доступности беспроводных точек доступа |
| iwconfig eth1 | показать конфигурацию беспроводного сетевого интерфейса eth1 |
| traceroute www.ya.ru | Трассировка маршрута до указанного хоста (www.ya.ru), аналог tracert в Windows. В некоторых дистрибутивах установлен по-умолчанию только traceroute6 и придётся доустанавливать вручную. |
| **Microsoft Windows networks(SAMBA)** | |
| nbtscan ip\_addr | разрешить netbios-имя *nbtscan не во всех системах ставится по-умолчанию, возможно, придётся доустанавливать вручную. nmblookup включен в пакет samba.* |
| nmblookup -A ip\_addr |
| smbclient -L ip\_addr/hostname | отобразить ресурсы, предоставленные в общий доступ на windows-машине |
| smbget -Rr smb://ip\_addr/share | подобно wget может получить файлы с windows-машин через smb-протокол |
| mount -t smbfs -o username=user,password=pass //winclient/share /mnt/share | смонтировать smb-ресурс, предоставленный на windows-машине, в локальную файловую систему |
| **IPTABLES (firewall)** | |
| iptables -t filter -nL | отобразить все цепочки правил |
| iptables -nL |
| iptables -t nat -L | отобразить все цепочки правил в NAT-таблице |
| iptables -t filter -F | очистить все цепочки правил в filter-таблице |
| iptables -F |
| iptables -t nat -F | очистить все цепочки правил в NAT-таблице |
| iptables -t filter -X | удалить все пользовательские цепочки правил в filter-таблице |
| iptables -t filter -A INPUT -p tcp --dport telnet -j ACCEPT | позволить входящее подключение telnet'ом |
| iptables -t filter -A OUTPUT -p tcp --dport http -j DROP | блокировать исходящие HTTP-соединения |
| iptables -t filter -A FORWARD -p tcp --dport pop3 -j ACCEPT | позволить "прокидывать" (forward) POP3-соединения |
| iptables -t filter -A INPUT -j LOG --log-prefix "DROP INPUT" | включить журналирование ядром пакетов, проходящих через цепочку INPUT, и добавлением к сообщению префикса "DROP INPUT" |
| iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE | включить NAT (Network Address Translate) исходящих пакетов на интерфейс eth0. Допустимо при использовании с динамически выделяемыми ip-адресами. |
| iptables -t nat -A PREROUTING -d 192.168.0.1 -p tcp -m tcp --dport 22 \  -j DNAT --to-destination 10.0.0.2:22 | перенаправление пакетов, адресованных одному хосту, на другой хост |
| **Мониторинг и отладка** | |
| top | отобразить запущенные процессы, используемые ими ресурсы и другую полезную информацию (с автоматическим обновлением данных) |
| ps -eafw | отобразить запущенные процессы, используемые ими ресурсы и другую полезную информацию (единожды) |
| ps -e -o pid,args --forest | вывести PID'ы и процессы в виде дерева |
| pstree | отобразить дерево процессов |
| kill -9 98989 | "убить" процесс с PID 98989 "на смерть" (без соблюдения целостности данных) |
| kill -KILL 98989 |
| kill -TERM 98989 | Корректно завершить процесс с PID 98989 |
| kill -1 98989 | заставить процесс с PID 98989 перепрочитать файл конфигурации |
| kill -HUP 98989 |
| lsof -p 98989 | отобразить список файлов, открытых процессом с PID 98989 |
| lsof /home/user1 | отобразить список открытых файлов из директории /home/user1 |
| strace -c ls > /dev/null | вывести список системных вызовов, созданных и полученных процессом ls |
| strace -f -e open ls > /dev/null | вывести вызовы бибилотек |
| watch -n1 'cat /proc/interrupts' | отображать прерывания в режиме реального времени |
| last reboot | отобразить историю перезагрузок системы |
| last user1 | отобразить историю регистрации пользователя user1 в системе и время его нахождения в ней |
| lsmod | вывести загруженные модули ядра |
| free -m | показать состояние оперативной памяти в мегабайтах |
| smartctl -A /dev/hda | контроль состояния жёсткого диска /dev/hda через SMART |
| smartctl -i /dev/hda | проверить доступность SMART на жёстком диске /dev/hda |
| tail /var/log/dmesg | вывести десять последних записей из журнала загрузки ядра |
| tail /var/log/messages | вывести десять последних записей из системного журнала |
| **Другие полезные команды** | |
| apropos *…keyword* | выводит список комманд, которые так или иначе относятся к ключевым словам. Полезно, когда вы знаете что делает программа, но не помните команду |
| man ping | вызов руководства по работе с программой, в данном случае, - ping |
| whatis *…keyword* | отображает описание действий указанной программы |
| mkbootdisk --device /dev/fd0 `uname -r` | создаёт загрузочный флоппи-диск |
| gpg -c file1 | шифрует файл file1 с помощью GNU Privacy Guard |
| gpg file1.gpg | дешифрует файл file1 с помощью GNU Privacy Guard |
| wget -r www.example.com | загружает рекурсивно содержимое сайта www.example.com |
| wget -c www.example.com/file.iso | загрузить файл www.example.com/file.iso с возможностью останова и продолжения в последствии |
| echo 'wget -c www.example.com/files.iso' | at 09:00 | начать закачку в указанное время |
| ldd /usr/bin/ssh | вывести список библиотек, необходимых для работы ssh |
| alias hh='history' | назначить алиас hh команде history |

History **Выводим список использованных команд (по у умолчанию сохраняется 1000 команд)**

## Полезная информация о системе и ядре

Перечислим некоторые важные файлы:

* /proc/cpuinfo - информация о процессоре ( модель, семейство, размер кэша и т.д.)
* /proc/meminfo - информация о RAM, размере свопа и т.д.
* sudo lshw -C memory вывести информацию по ОЗУ
* Dmidecode - вся информация по железу (проц, память и т.д)
* # hdparm -I /dev/sda расширенная информация по жёсткому диску
* # lshw -class disk -class storage расширенная информация по жёсткому диску
* # lshw -short -C disk информация об установленных дисках в системе
* /proc/mounts - список подмонтированных файловых систем.
* /proc/devices - список устройств.
* /proc/filesystems - поддерживаемые файловые системы.
* /proc/modules - список загружаемых модулей.
* /proc/version - версия ядра.
* /proc/cmdline - список параметров, передаваемых ядру при загрузке.
* cat /etc/\*-release - узнать версию дистрибутива Linux
* uname –a - определение какая разрядность у операционной системы
* arch -- В окне терминала также отобразится разрядность операционной системы
* echo $LANG  **-** Узнать, какой язык настроен в вашей системе
* sed 's/:.\*//' /etc/passwd - если вам нужны только имена пользователей.
* w - какие пользователи сейчас активны в системе, и какие команды они выполняют.
* sudo service **ssh** stop|start|restart - При установке SSH-сервер автоматически прописывается в автозагрузку. Управлять его запуском, остановкой или перезапуском можно с помощью команд
* echo $(uptime) - Вывести информацию о загрузке и числа подключённых пользователях.
* echo $? - Командная оболочка поддерживает переменную, посредством кото­рой можно определить код завершения. 0 служит признаком успешного завершения, а любое другое нет
* id -u - выведет числовой идентификатор действующего пользователя. Суперполь­зователю всегда присваивается числовой идентификатор 0.
* Update-manager - проверить есть ли обновления системы. Для Ubuntu.
* service --status-all Для вывода списка всех запущенных сервисов и просмотра их статусов
* cat /proc/номер PID в команде ps aux/status - посмотреть как именно ядро выделило память этому процессу.
* systemd-cgls - команда выдаёт иерархию выделения ресурсов в сгрупированном виде запущенных приложений.
* apt-get install open-vm-tools –y поставить драйвера на виртуалку для ESXI . Чтобы потом проверить статус /etc/init.d/ open-vm-tools status
* /var/log/dmesg ядро фиксирует в этом журнале информацию о найденных устройствах и драйверах, если таковые у него нашлись.
* Lspci Эта программа показывает информацию обо всех устройствах, которые подключены к системной шине, и которые удалось распознать.
* Lsblk посмотреть какие у меня разделы имеются
* lshw -class disk -class storage Эта программа показывает информацию о HDD
* smartctl -d ata -a -i /dev/sda или smartctl -x /dev/sda При помощи программы smartctl можно выводить информацию о состоянии ваших устройств, которые поддерживают SMART
* blkid - отобразит все диски с UUID и типом файл. системы.
* pvdisplay - Посмотреть физические тома
* vgdisplay - Показать volume группу
* lvdisplay - Посмотреть логические тома в системе

sudo apt-get install –f если в системе есть сообщения о том что нужно установить зависимости.

sudo mount dev/sdf2 /media/usb0 -o rw,uid=$(id -u)

**Монтирование внешнего жёсткого диска**

cat /etc/passwd | awk '/bash/{print}' Посмотреть всех пользователей

Список всех сервисов и их статус можно посмотреть с помощью утилиты systemctl, для этого наберите в терминале (**Ctrl+Alt+T**):

systemctl list-units --type=service

А чтобы посмотреть список сервисов, которые сейчас выполняются выполните:

systemctl list-units --type=service --state=running

Аналогичным образом можно посмотреть сервисы, которые уже добавлены в автозагрузку:

systemctl list-unit-files --type=service --state=enabled

Добавить сервис в автозагрузку можно с помощью утилиты опции enable:

sudo systemctl enable имя\_сервиса

Удалить сервис из автозагрузки можно указав опцию disable:

sudo systemctl disable имя\_сервиса

## Настройка менеджера пакетов

Ограничиваем количество установленных версий пакетов до 3-х (например количество версий установленного ядра):

|  |  |
| --- | --- |
| 1  2  3 | $ nano -w /etc/yum.conf  installonly\_limit=3  $ yum -y install yum-cron yum-utils |

Включаем проверку новых версий пакетов (информация о новых версиях будет приходить по почте, сами пакеты обновляться автоматически не будут):

|  |  |
| --- | --- |
| 1  2  3  4 | $ nano -w /etc/yum/yum-cron.conf  download\_updates = no  $ systemctl enable yum-cron.service  $ systemctl start yum-cron.service |

## Установка необходимых утилит и программ

Теперь подключаем EPEL-репозиторий (Extra Packages for Enterprise Linux) и доустанавливаем необходимые пакеты, отсутствующие в базовой установке:

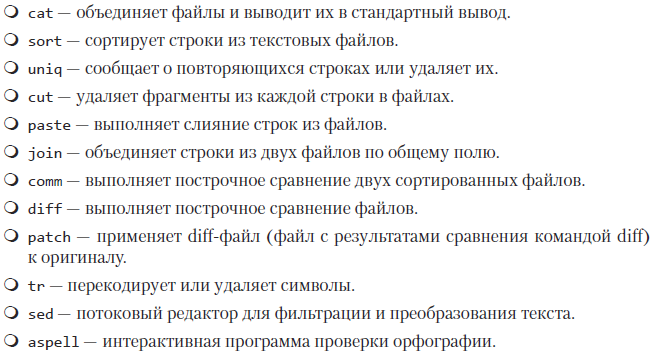
|  |  |
| --- | --- |
| 1  2  3  4 | $ rpm -Uvh http://mirror.yandex.ru/epel/7/x86\_64/e/epel-release-7-2.noarch.rpm  $ rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7  $ yum -y install bash-completion logwatch lsscsi mc mlocate mtr nfs-utils parted pciutils \           pinfo policycoreutils-python rsync screen sysstat telnet unzip usbutils wget |

Есть ещё интересный пакет atop, в котором можно посмотреть статистику загрузки сервера в реальном времени с разбивкой по процессам и устройствам. Но в настоящий момент этот пакет для CentOS 7 находится в тестовом репозитории EPEL, поэтому устанавливается он следующей командой:

|  |  |
| --- | --- |
| 1 | $ yum -y install atop --enablerepo=epel-testing |

Update: atop уже поместили в основной репозиторий, поэтому его можно устанавливать просто командой:

|  |  |
| --- | --- |
| 1 | $ yum -y install atop |
|  |  |



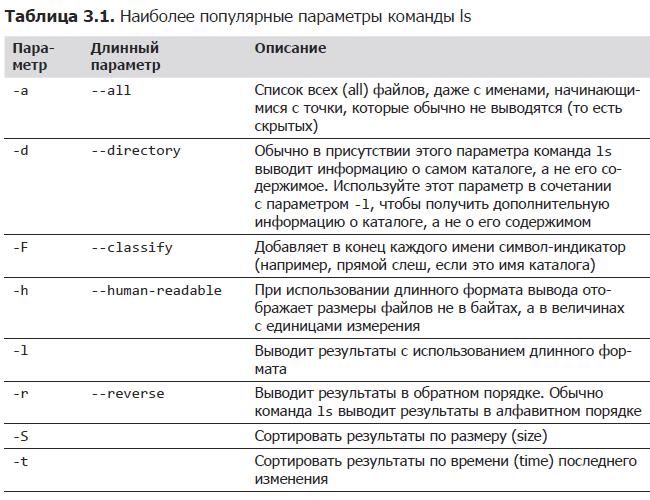
Увидеть региональные настройки можно, выполнив команду locale

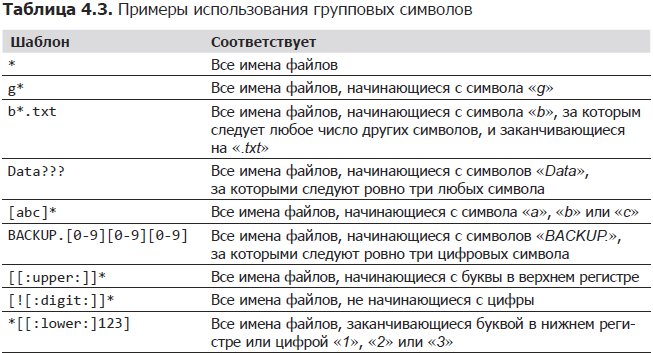
[me@linuxbox ~]$ **locale**

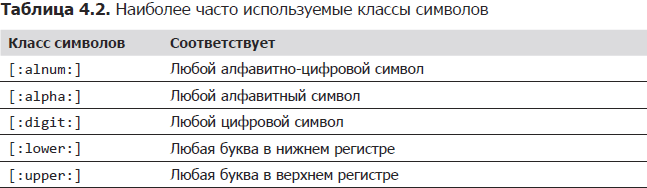
Чтобы установить региональные настройки, обеспечивающие традиционное поведение системы Unix, присвойте переменной LANG значение POSIX:

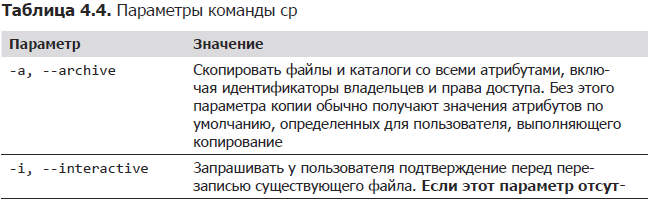
[me@linuxbox ~]$ **export LANG=POSIX**

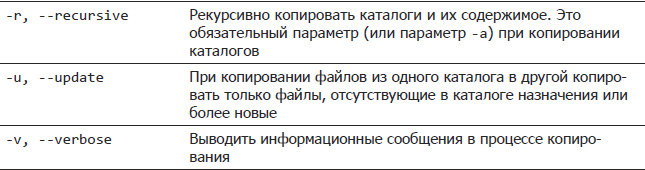
Имейте в виду, что в результате наших действий система будет использовать набор символов американского английского

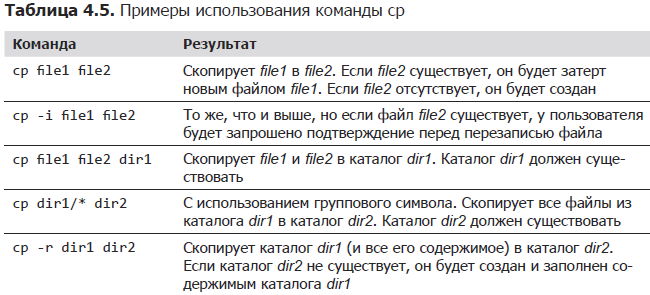


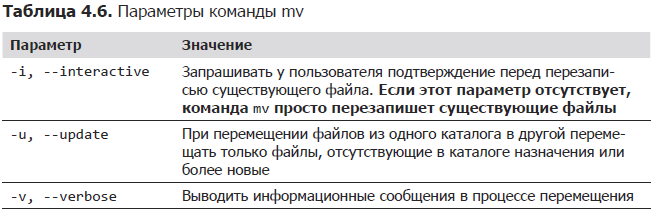


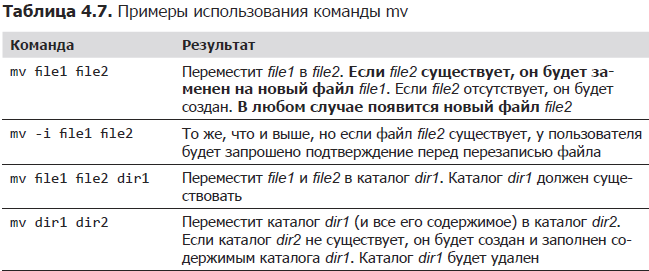


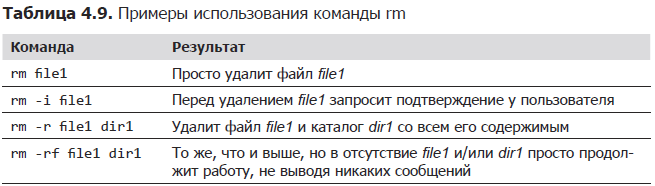


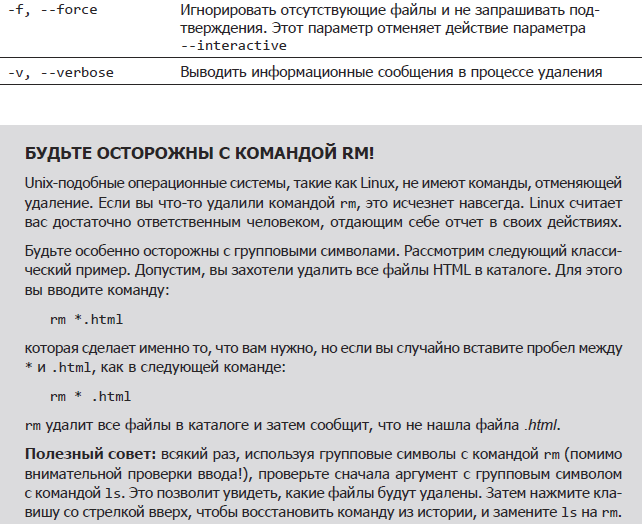
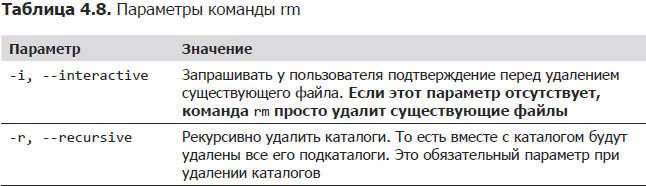


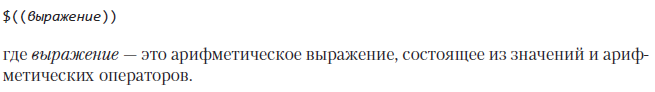
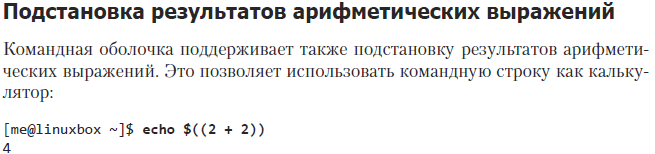


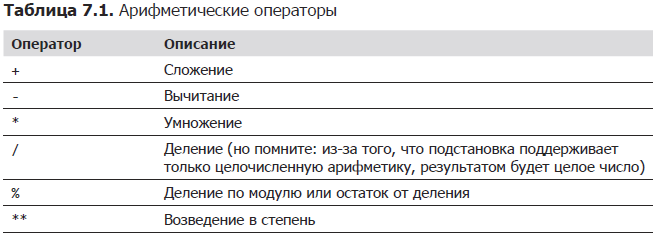


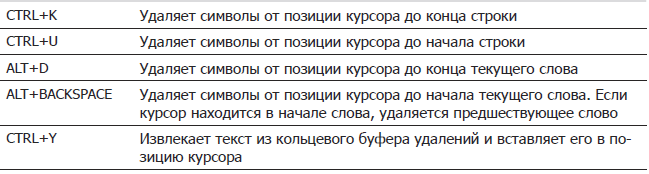
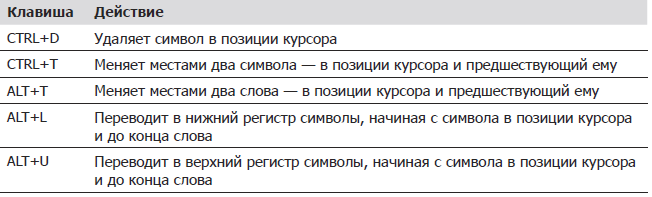


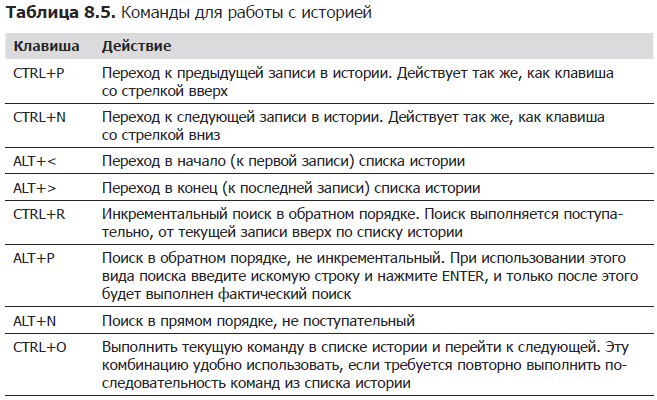


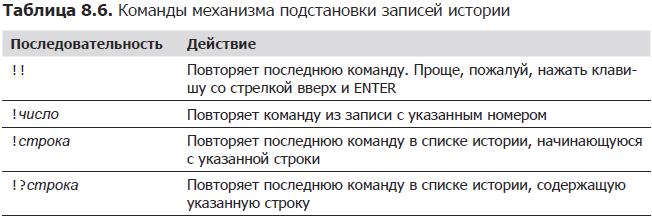


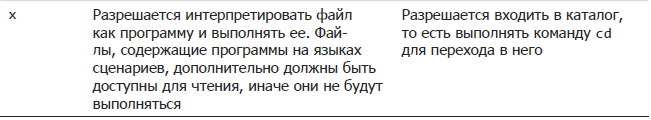
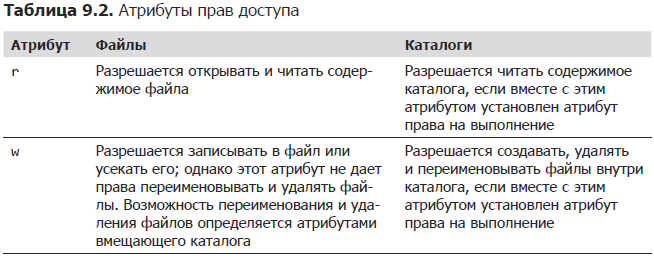
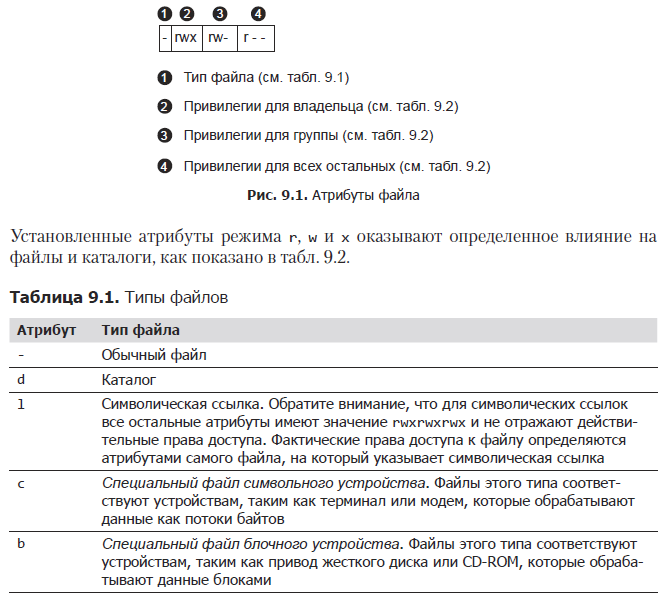


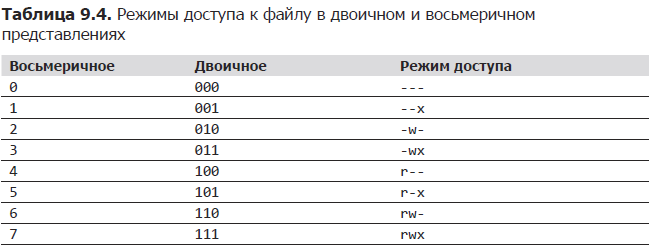






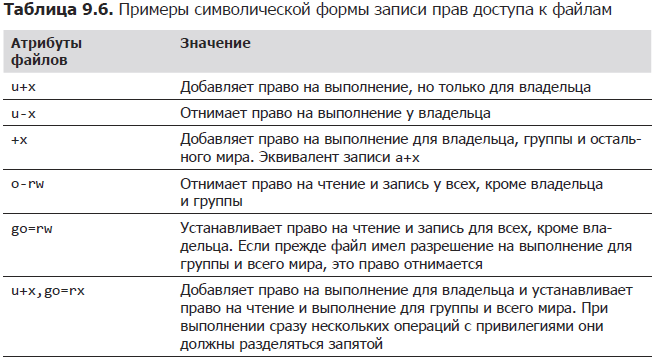
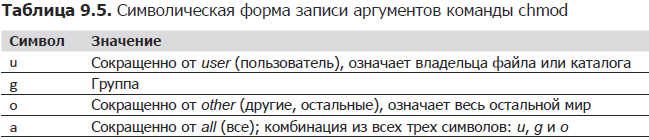






Пример





Пример : chmod u+x foo.txt

**-----------------------------------------------------------------------------------**

**umask — определение разрешений доступа к файлам по умолчанию**

Команда umask определяет разрешения по умолчанию, которые устанавливаются для файла при его создании.

Далее мы выполнили команду umask без аргумента, чтобы увидеть текущее значение маски. Она вернула нам значение 0002 (часто также использует­ся значение 0022) — восьмеричное представление действующей маски.

Если посмотреть на значение маски 0022, легко увидеть, что оно делает:

|  |  |
| --- | --- |
|  |  |
| Маска | 000 000 010 010 |
| Результат | --- rw- r-- r-- |

Команда -- Umask 0022

Дополнительно в книге Командная строка Linux. Полное руководство.pdf стр. 111

**-----------------------------------------------------------------------------------**

**Чтобы увидеть, какие привилегии дает команда sudo, вызовите ее с параметром -l:**

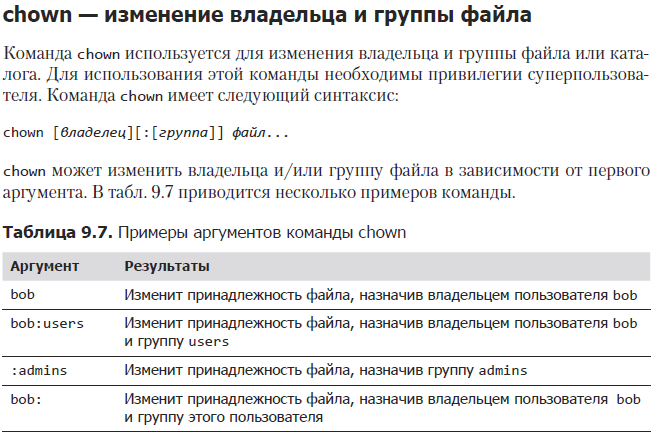
**[me@linuxbox ~]$ sudo -l**

**User me may run the following commands on this host:**

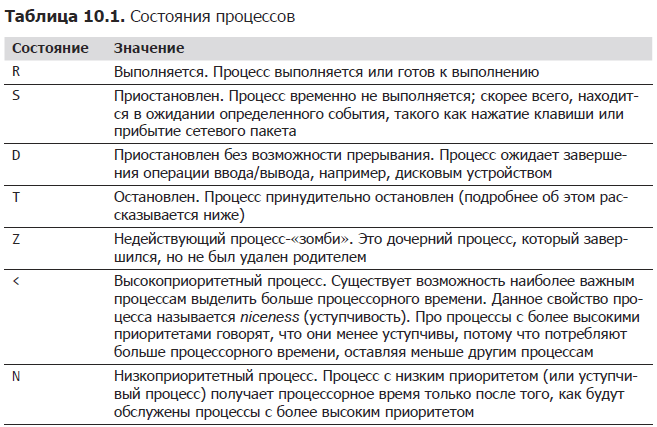
**(ALL) ALL**

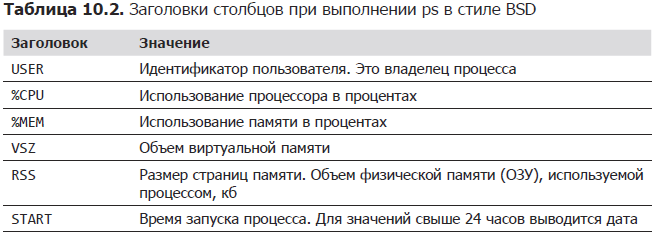
**-----------------------------------------------------------------------------------**

**chown**

****

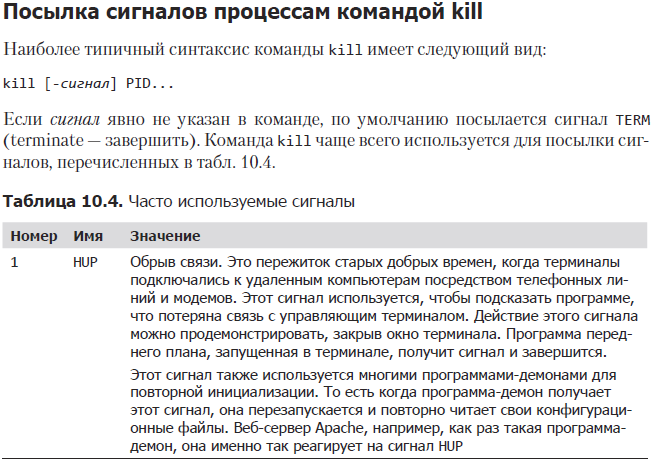
**-----------------------------------------------------------------------------------**

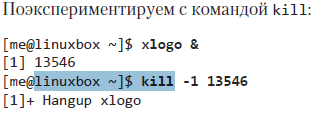
****

****

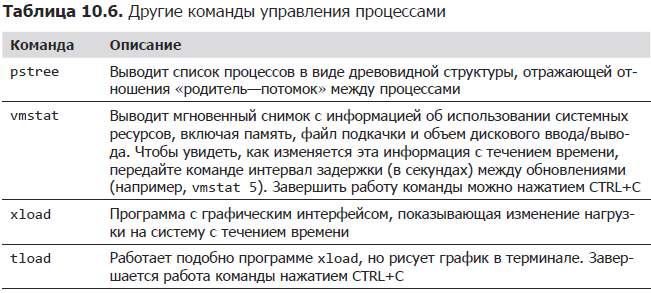
**-----------------------------------------------------------------------------------**

**KILL**

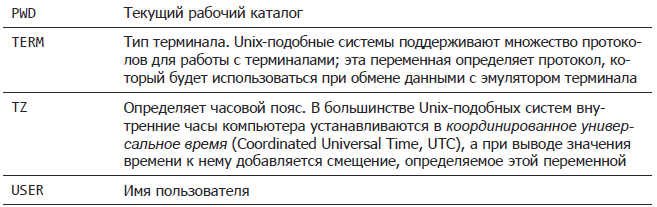
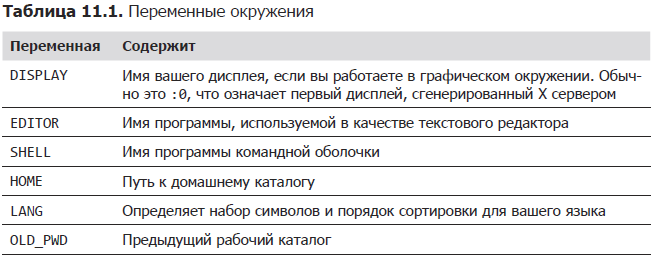
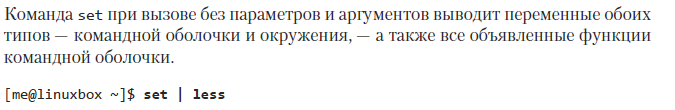
****

****

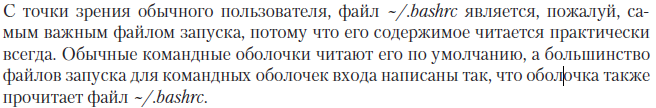
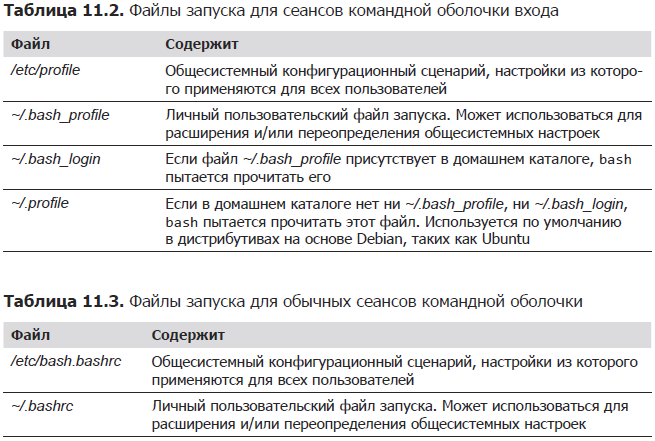
**---------------------------------------------------------------**

****

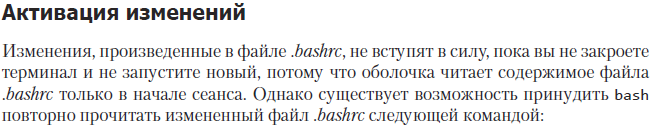
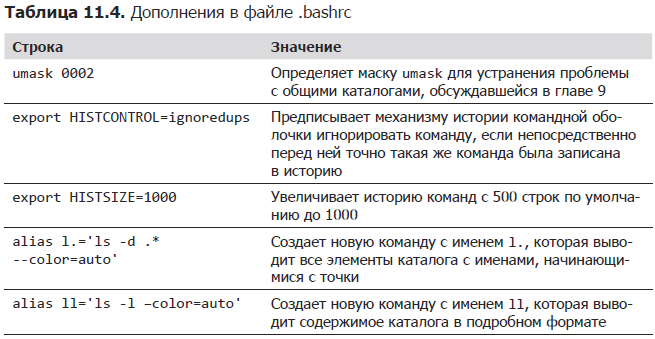
**-----------------------------------------------------------------------------------**

****

**----------------------------------------------------------------------------------------**

****

**--------------------------------------------------------------------------**

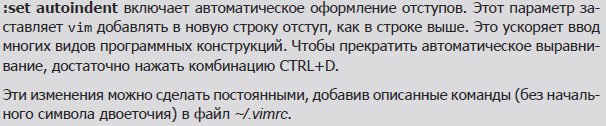
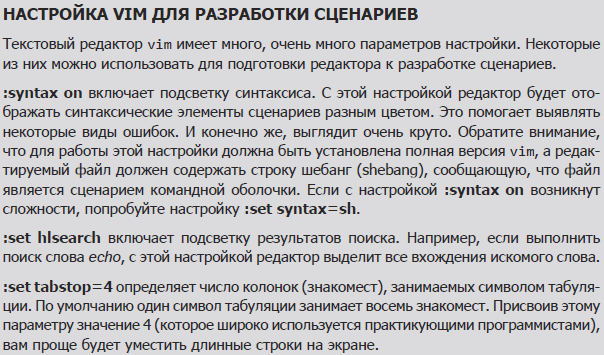
****

****

**-------------------------------------------------------------------------------------**

****

**и VIM**

****

:set number VIM: отобразить номера строк

Что бы выключить отображение – выполняем:

: set nu!

Что бы включить нумерацию строк в редакторе vim по-умолчанию для всех файлов – отредактируйте файл ~/.vimrc, и в него добавьте строку:

set number

Например так:

|  |  |
| --- | --- |
| 1 | echo "set number" >> .vimrc |

И небольшое дополнение:

Открыть файл и сразу перейти к строке 200:

|  |  |
| --- | --- |
| 1 | $ vim +200 server.xml |

Для этого введите следующую команду (об­ратите внимание: двоеточие — это часть команды):

**:q**

В окне терминала должно появиться приглашение к вводу командной оболочки. Если по какой-то причине выход из vi не получился (скорее всего, потому что вы внесли какие-то изменения и еще не сохранили их), сообщите vi, что вы действи­тельно хотите выйти, добавив в команду восклицательный знак:

**:q!**

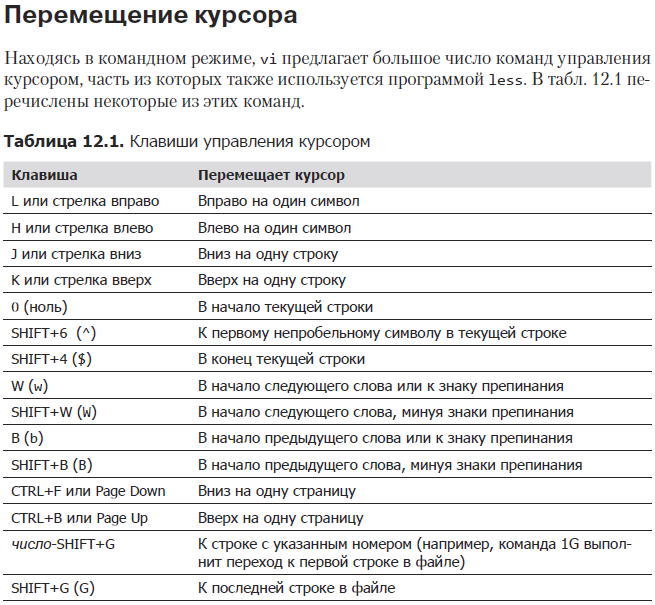
Чтобы добавить какой-то текст в файл, необходимо сначала перейти в *режим вставки*. Для этого нажмите клавишу I (i).

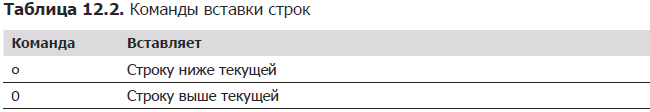
**Сохранение изменений**

Чтобы выполнить запись изменений в файл, вслед за двоеточием введите w и на­жмите ENTER:

:w

Сохранить и выйти из файла (аналог :wq) --- :x





Если нажать клавишу U в командном режиме, vi отменит последнее вы­полненное изменение. Это пригодится нам, когда мы будем пробовать некоторые простые команды редактирования.

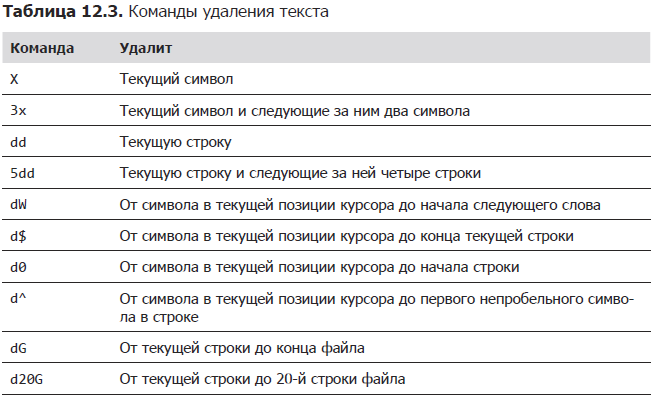
**Добавление текста в конец**

vi поддерживает команду добавления текста в конец, разумно названную a. Если переместить курсор в конец строки и ввести a, курсор переместится за конец стро­ки и vi перейдет в режим вставки.

Поскольку добавлять текст в конец строки требуется довольно часто, vi предла­гает сокращенную команду для перемещения в конец строки и перехода в режим добавления. Это команда A.

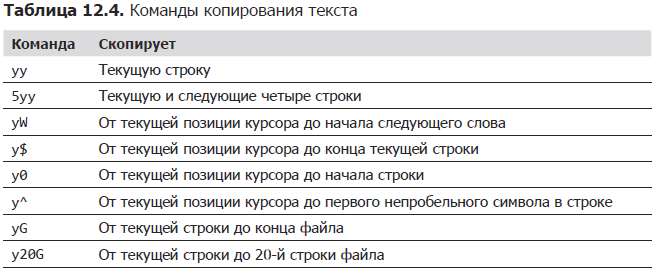
**Удаление текста**

Первый способ: клавиша X удаляет символ в позиции курсора. Команде x может предшествовать число, определяю­щее количество удаляемых символов.



Команда d не просто удаляет текст, она «вырезает» его, позднее его можно извлечь командой p и вставить правее позиции курсора или левее — командой P.

Команда y выполняет копирование (yank) текста в буфер вставки почти так же, как команда d.



**Объединение строк**

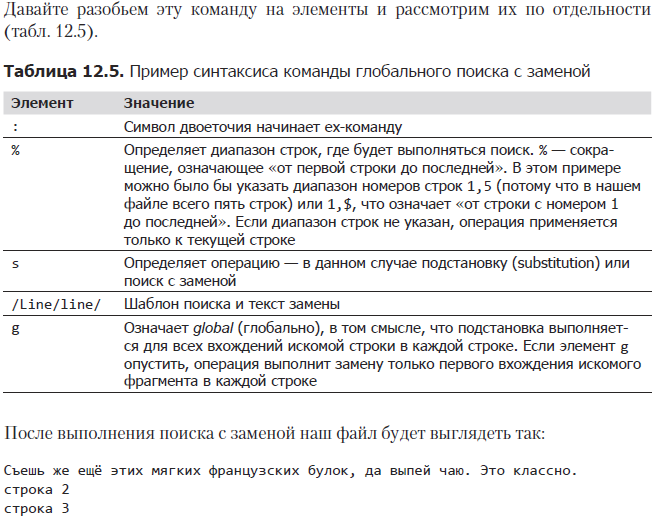
в vi была добавлена специальная команда J (не путайте с командой j, которая перемещает курсор на одну строку вниз) для объединения строк.

**Поиск во всем файле**

После ввода команды / в нижней части экрана появится прямой слеш, вслед за которым нужно ввести искомое слово или фразу и нажать ENTER. Затем введите команду n, и курсор переместится в строку. С каждой следующей командой n курсор будет пе­ремещаться вниз по файлу, пока не достигнет последнего вхождения.

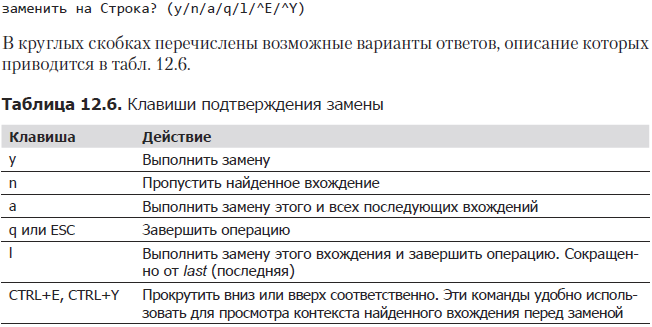
**Глобальный поиск и замена**

:%s/Строка/строка/g



В команде подстановки можно указать, что она должна запрашивать подтвержде­ние у пользователя перед заменой. Для этого добавьте символ c в конец команды. Например:

:%s/строка/Строка/gc



**Редактирование нескольких файлов**

Теперь откройте в vi старый и новый файлы:

Пример:

[me@linuxbox ~]$ **vi foo.txt ls-output.txt**

**Переключение между файлами**

Чтобы переключиться с одного файла на следующий, выполните ex-команду:

**:n**

Чтобы вернуться обратно, в предыдущий файл, выполните:

**:N**

командой :buffers можно вы­вести список редактируемых файлов.

Также существует возможность добавлять файлы в текущий сеанс редактирова­ния. Команда :e (сокращенно от *edit* — редактировать) с именем файла откроет дополнительный файл.

Файлы, открытые командой :e, недоступны для команд :n и :N. Чтобы переключиться на такие файлы, используйте команду :buffer с номером буфера.

**---------------------------------------------------------------------------------------------------**

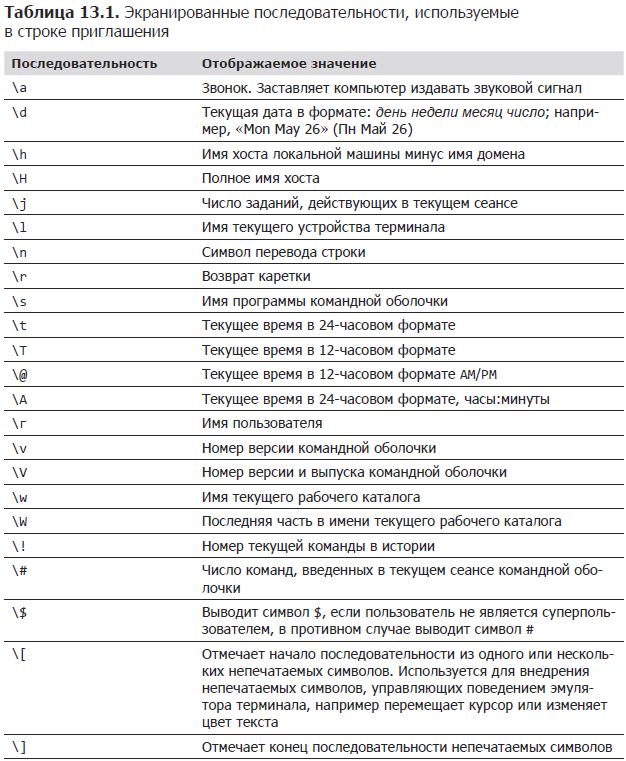
**Устройство строки приглашения к вводу**

По умолчанию строка приглашения к вводу имеет следующий вид:

[me@linuxbox ~]$

[me@linuxbox ~]$ **echo $PS1**

[\u@\h \W]\$



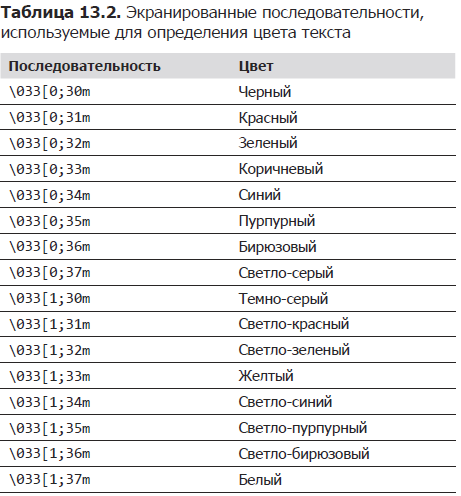
**Добавление цвета**

Давайте попробуем окрасить строку приглашения в красный цвет (здесь она вы­глядит как серая). Добавьте в начало экранированную последовательность:

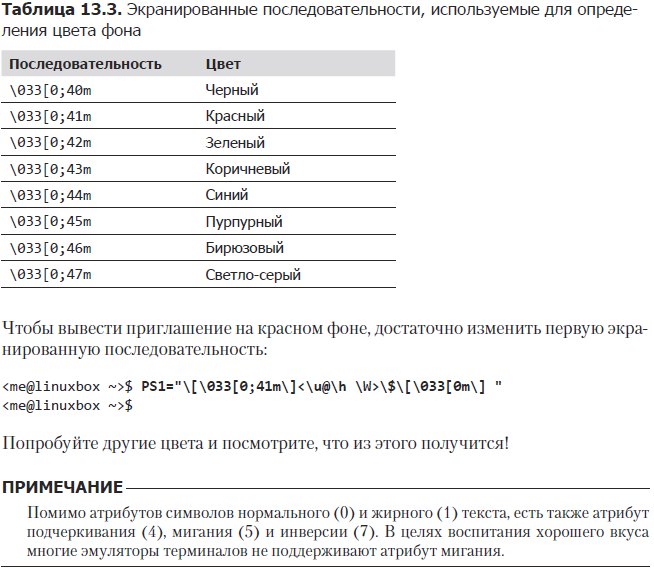
<me@linuxbox ~>$ **PS1="\[\033[0;31m\]<\u@\h \W>\$ "**

Получилось, но обратите внимание, что весь текст, который вводится с клавиату­ры вслед за строкой приглашения, также окрашивается в красный цвет. Для устра­нения этого эффекта добавьте еще одну экранированную последовательность

<me@linuxbox ~>$ **PS1="\[\033[0;31m\]<\u@\h \W>\$\[\033[0m\] "**

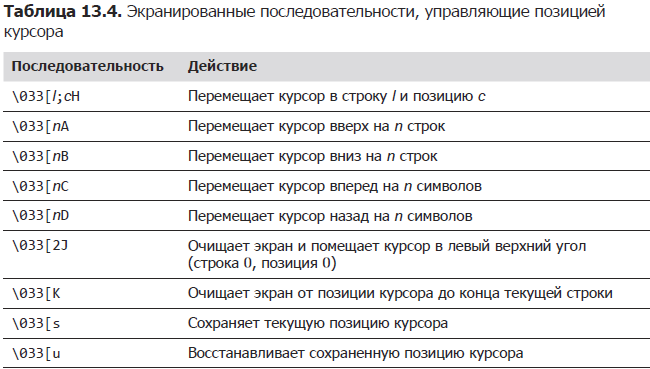


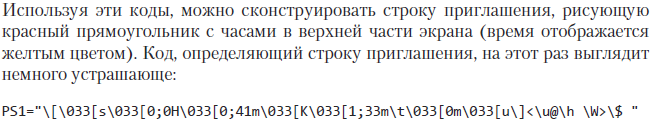
Кроме того, существует возможность изменить цвет фона. Цвет фона не поддерживает атрибут жирного текста.

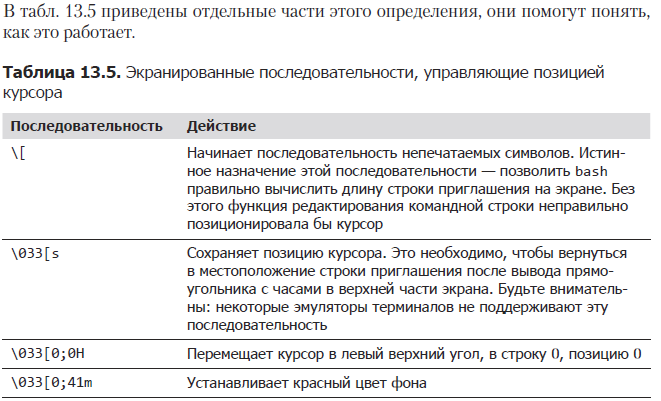


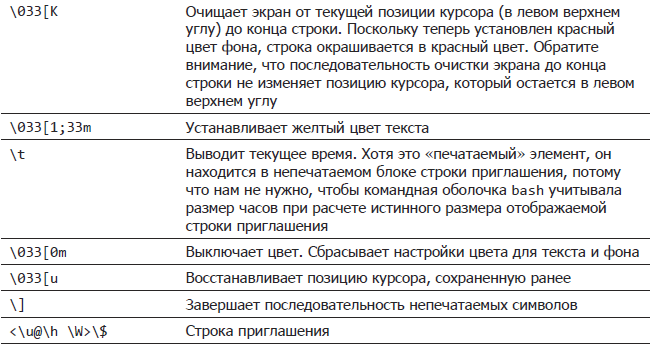
**Перемещение курсора**

Экранированные последовательности можно использовать для позиционирова­ния курсора. Этот прием часто используется для отображения времени или дру­гой информации в разных местах на экране, например вверху, при каждом выводе приглашения к вводу



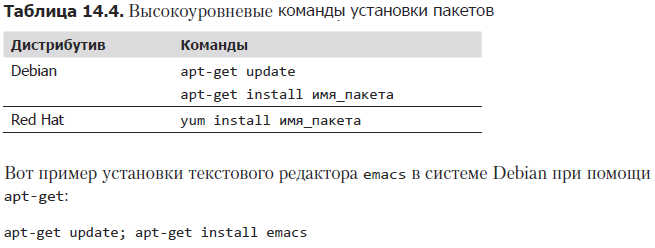




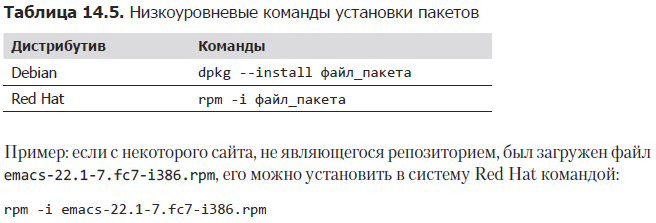


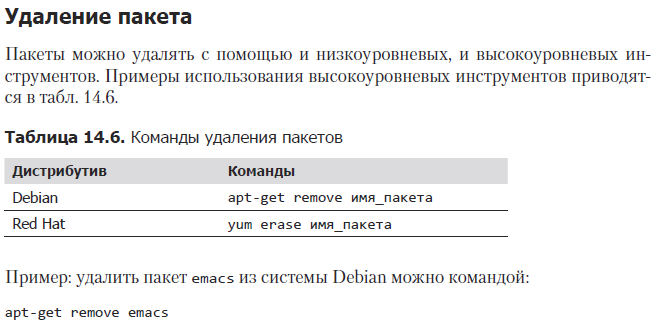
------------------------------------------------------------------------------

**Установка пакета из репозитория**

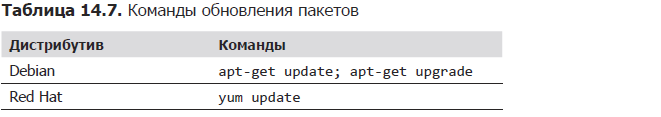


**Установка пакета из файла пакета**

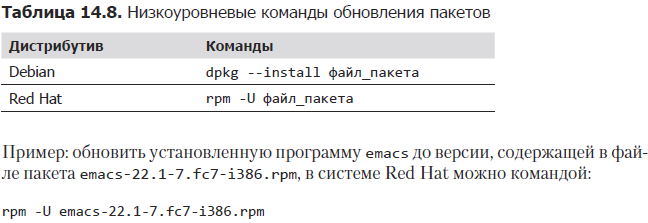


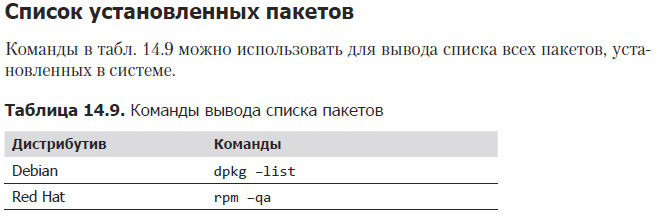


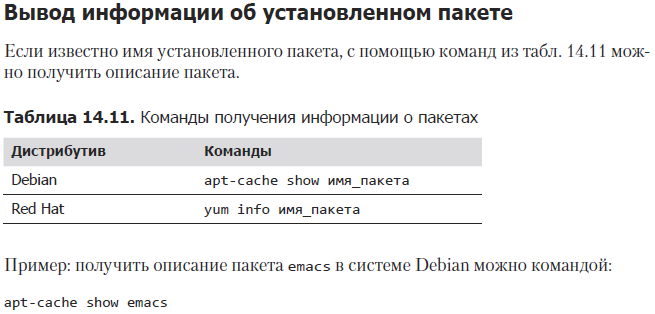
**Обновление пакетов из репозитория**



**Обновление пакета из файла пакета**







------------------------------------------------------------------

**Монтирование и размонтирование устройств хранения**

В файле с именем */etc/fstab* перечисляются устройства (обычно разделы жестко­го диска), монтируемые на этапе загрузки.

**Просмотр списка смонтированных файловых систем**

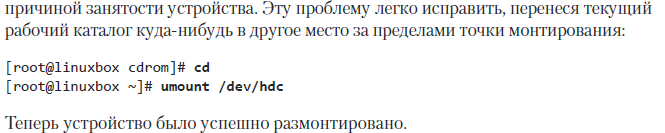
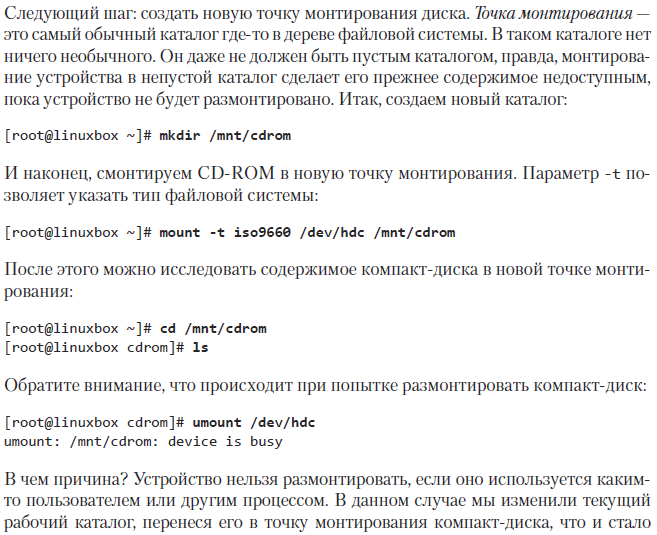
[me@linuxbox ~]$ **mount**

/dev/sda2 on / type ext3 (rw)

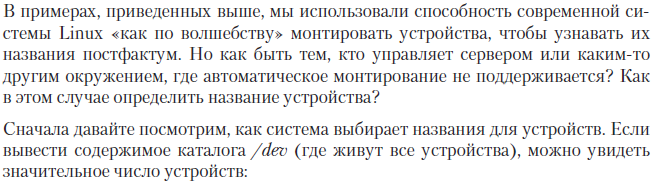
proc on /proc type proc (rw)

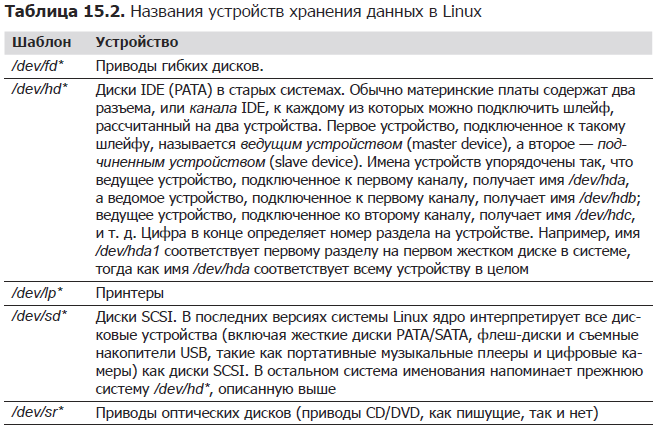
sysfs on /sys type sysfs (rw)

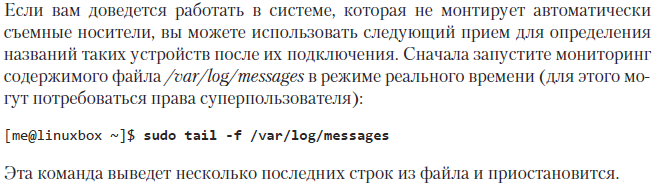
Список имеет следующий формат: *устройство* on *точка\_монтирования* type *тип\_файловой\_системы* (*параметры*). Например, первая строка соответствует устройству */dev/sda2*, смонтированному как корневая файловая система типа ext3, доступная для чтения и записи (параметр rw).



**Определение названий устройств**







Когда вывод опять приостановится, нажмите CTRL+C, чтобы вернуться в пригла­шение командной строки. Наибольший интерес для нас представляют строки с упоминанием имени устройства [sdb], соответствующего нашим ожиданиям в отношении названия устройства диска SCSI.

Они сообщают, что имя */dev/sdb* соответствует всему устройству, а имя */dev/sdb1* — первому разделу на этом устройстве.

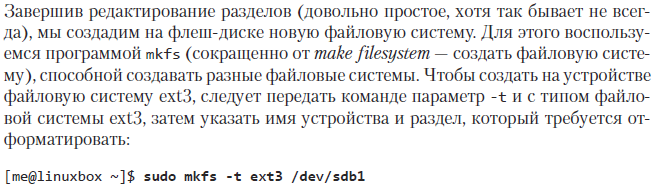
Зная имя устройства, можно смонтировать флеш-диск:

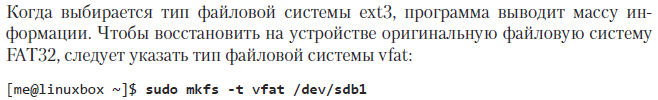
[me@linuxbox ~]$ **sudo mkdir /mnt/flash**

[me@linuxbox ~]$ **sudo mount /dev/sdb1 /mnt/flash**

[me@linuxbox ~]$ **df -H**

**Создание новой файловой системы с помощью mkfs**





-----------------------------------------------------

**Непосредственное перемещение данных между устройствами**

Она копирует блоки данных из одного места в другое. По историческим причинам команда имеет уникальный синтаксис:

**dd if=*входной\_файл* of=*выходной\_файл* [bs=*размер\_блока* [count=*число\_блоков*]]**

Допустим, что после подключе­ния к компьютеру им назначаются имена устройств */dev/sdb* и */dev/sdc* соответ­ственно. В этом случае скопировать содержимое первого диска на второй можно следующей командой:

**dd if=/dev/sdb of=/dev/sdc**

Как вариант, если к компьютеру подключено только первое устройство, можно скопировать его содержимое в обычный файл, который впоследствии использо­вать для восстановления или копирования:

**dd if=/dev/sdb of=flash\_drive.img**

-------------------------------------------------------

**Создание образа компакт-диска**

Запись на компакт-диски (CD-R или CD-RW) выполняется в два этапа: сначала нужно создать файл образа ISO, являющийся точным образом файловой системы компакт-диска, а затем записать файл образа на носитель (то есть на сам компакт-диск).

**Создание образа-копии компакт-диска**

Чтобы создать ISO-образ имеющегося компакт-диска, необходимо с помощью dd прочитать все блоки с данными с этого компакт-диска и скопировать их в ло­кальный файл. Например, допустим, что у нас есть компакт-диск с дистрибути­вом Ubuntu, и мы хотим создать файл ISO-образа, который потом можно будет использовать для создания нескольких копий. Вставив компакт-диск в привод CD-ROM и определив имя устройства (пусть это будет */dev/cdrom*), мы сможем создать файл ISO-образа следующим способом:

**dd if=/dev/cdrom of=ubuntu.iso**

Этот прием также применим к дискам DVD с данными, но он не будет работать с аудиодисками, так как для хранения данных на них файловая система не ис­пользуется. Если вы хотите скопировать аудиодиск, обратитесь к команде cdrdao.

**Создание образа из коллекции файлов**

Создать файл ISO-образа, включающий содержимое некоего каталога, можно с помощью программы genisoimage. Для этого сначала создадим каталог со все­ми необходимыми файлами для включения в образ и затем командой genisoimage создадим файл образа. Например, если предположить, что вы создали каталог *~/cd-rom-files* и наполнили его файлами для записи на компакт-диск, следующая команда создаст файл образа с именем *cd-rom.iso*:

**genisoimage -o cd-rom.iso -R -J ~/cd-rom-files**

**Непосредственное монтирование файла ISO-образа**

Существует один трюк, позволяющий монтировать ISO-образы, хранящиеся на жестком диске, и работать с ними, как если бы это были оптические носители. Па­раметр -o loop, добавленный в команду mount (вместе с обязательным параметром -t iso9660, определяющим тип файловой системы), позволяет смонтировать файл образа в дерево файловой системы, как если бы это было обычное устройство:

**mkdir /mnt/iso\_image**

**mount -t iso9660 -o loop image.iso /mnt/iso\_image**

**Очистка перезаписываемых компакт-дисков**

Перезаписываемые компакт-диски CD-RW нужно стирать, или очищать, перед повторным использованием. Для этого воспользуемся командой wodim, указав ей имя устройства пишущего привода компакт-дисков и тип очистки. Программа wodim предлагает несколько типов очистки. Для минимальной (и самой быстрой) очистки следует указать тип fast:

**wodim dev=/dev/cdrw blank=fast**

**Запись образа**

Записать образ можно с помощью все той же программы wodim, указав ей имя устройства пишущего привода компакт-дисков и имя файла образа:

**wodim dev=/dev/cdrw image.iso**

Помимо имени устройства и файла образа программа wodim поддерживает массу дополнительных параметров. Чаще других используются параметры -v (обеспе­чивает вывод подробной информации в ходе записи) и -dao (выполняет запись на диск в режиме *disc-at-once* — диск целиком). Режим «диск целиком» следует ис­пользовать, если вы собираетесь воспроизводить диски в коммерческих целях. По умолчанию wodim использует режим *track-at-once* (по одной дорожке).

Часто бывает полезно проверить целостность ISO-образа, распространители ISO-образов сопровождают их *файлами с контрольными суммами*. Контрольная сумма — это результат экзоти­ческих математических вычислений в виде числа, представляющего содержимое целевого файла. Для вычисления контрольной суммы чаще всего используется программа md5sum, воз­вращающая уникальное шестнадцатеричное число:

**md5sum image.iso**

34e354760f9bb7fbf85c96f6a3f94ece image.iso

Многие типы носителей, такие как DVD, требуют точного вычисления числа бло­ков. Следующий пример демонстрирует проверку целостности файла образа *dvd-image.iso* и диска в устройстве */dev/dvd* привода DVD.

**md5sum dvd-image.iso; dd if=/dev/dvd bs=2048 count=$(( $(stat -c "%s" dvd-image.iso) / 2048 )) | md5sum**

-----------------------------------------------------------------------------

**traceroute — трассировка пути сетевых пакетов**

Программа traceroute (в некоторых системах используется похожая на нее про­грамма tracepath) выводит список всех «переходов» (hops) на пути сетевого тра­фика между локальной системой и указанным узлом сети.

[me@linuxbox ~]$ **traceroute slashdot.org**

**netstat — вывод параметров настройки сети и статистик**

Программа netstat используется для исследования различных настроек сети и статистик. С помощью множества параметров этой команды можно просматри­вать самые разные аспекты настройки сети. С помощью параметра -ie, например, можно исследовать сетевые интерфейсы в системе:

[me@linuxbox ~]$ **netstat –ie**

Подобная команда: **ifconfig**

Использование параметра -r позволит получить таблицу маршрутизации ядра. По этой таблице можно судить, как настроена передача пакетов между сетями:

[me@linuxbox ~]$ **netstat –r**

Подобная команда: route –v

-----------------------------------------

**ftp — передача файлов по протоколу FTP**

[me@linuxbox ~]$ **ftp fileserver**

Connected to fileserver.localdomain.

220 (vsFTPd 2.0.1)

Name (fileserver:me): **anonymous**

331 Please specify the password.

Password:

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> **cd pub/cd\_images/Ubuntu-8.04**

250 Directory successfully changed.

ftp> **ls**

200 PORT command successful. Consider using PASV.

150 Here comes the directory listing.

-rw-rw-r-- 1 500 500 733079552 Apr 25 03:53 ubuntu-8.04-desktopi386.iso

226 Directory send OK.

ftp> **lcd Desktop**

Local directory now /home/me/Desktop

ftp> **get ubuntu-8.04-desktop-i386.iso**

local: ubuntu-8.04-desktop-i386.iso remote: ubuntu-8.04-desktop-i386.iso

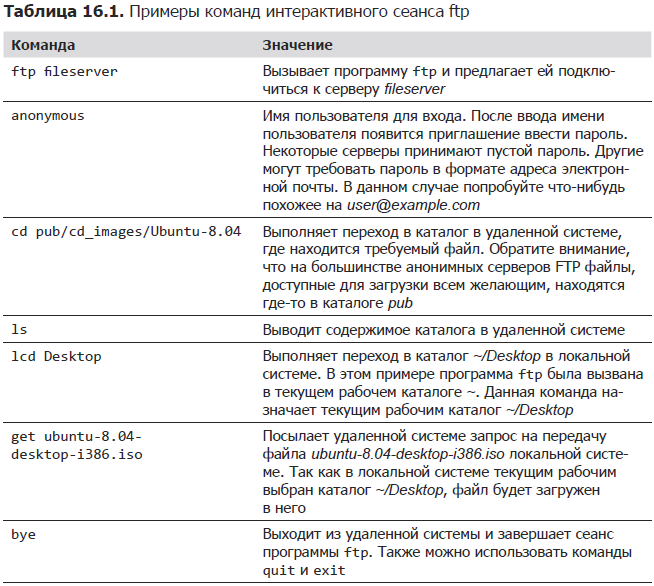
200 PORT command successful. Consider using PASV.

150 Opening BINARY mode data connection for ubuntu-8.04-desktop-i386.iso

(733079552 bytes).

226 File send OK.

733079552 bytes



**scp и sftp — безопасная передача файлов**

Первая, scp (secure copy — без­опасное копирование) используется для копирования файлов, как уже знакомая вам программа cp. Основное отличие заключается в необходимости предварять пути к исходному или конечному файлу именем удаленного узла и символом двоеточия за ним.

[me@linuxbox ~]$ **scp remote-sys:document.txt .**

me@remote-sys's password:

document.txt 100% 5581 5.5KB/s 00:00

[me@linuxbox ~]$

Вторая программа копирования файлов через SSH-соединение: sftp. Как следует из ее имени — это безопасная замена для программы ftp. sftp действует практи­чески так же, как оригинальная программа ftp

[me@linuxbox ~]$ **sftp remote-sys**

Connecting to remote-sys...

me@remote-sys's password:

sftp> **ls**

ubuntu-8.04-desktop-i386.iso

sftp> **lcd Desktop**

sftp> **get ubuntu-8.04-desktop-i386.iso**

Fetching /home/me/ubuntu-8.04-desktop-i386.iso to ubuntu-8.04-desktop-i386.iso

/home/me/ubuntu-8.04-desktop-i386.iso 100% 699MB 7.4MB/s 01:35

sftp> **bye**

---------------------------------------------------------------------

**Поиск файлов**

locate – выполняет поиск файлов по именам.

find – выполняет поиск файлов в иерархии каталогов.

xargs – конструирует команды на основе данных, полученных из стандартного ввода, и выполняет их.

В современных дистрибутивах Linux, — это slocate и mlocate, ко­торые, впрочем, являются символическими ссылками, указывающими на locate.

В некоторых дистрибутивах при попытке запустить locate сразу после установки она потерпит неудачу, но если попытаться использовать ее на следующий день, все, как ни странно будет работать как надо. Поскольку база данных не обновляется непрерывно, скорее всего, locate не находит самые свежие файлы. Чтобы решить эту проблему, запустите программу updatedb вручную от имени суперпользователя.

[me@linuxbox ~]$ **locate bin/zip**

Если к результатам поиска предъявляются более строгие требования, команду locate можно объединить с другими инструментами, такими как grep, позволяю­щими осуществить более сложный поиск:

[me@linuxbox ~]$ **locate zip | grep bin**

**find — сложный способ поиска файлов**

Например, с ее помощью можно получить список содержи­мого домашнего каталога:

[me@linuxbox ~]$ **find ~**

Её можно передать по конвейеру другим программам. Воспользуемся программой wc, чтобы подсчитать число файлов:

[me@linuxbox ~]$ **find ~ | wc -l**

47068

Допустим, мы хотим получить список каталогов. Для этого добавим в команду следующую проверку

[me@linuxbox ~]$ **find ~ -type d | wc -l**

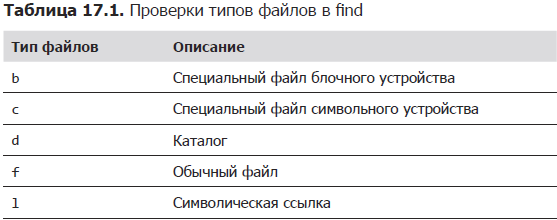
1695

Добавив проверку -type d, мы ограничились поиском только каталогов.

Но точно так же можно ограничить поиск только обычными файлами:

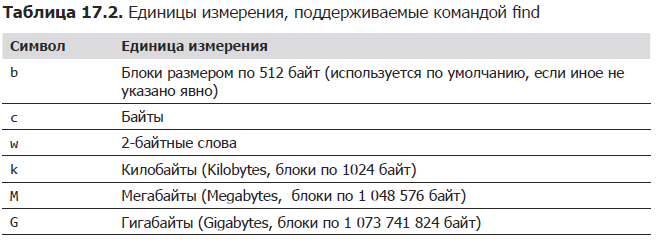
[me@linuxbox ~]$ **find ~ -type f | wc -l**

38737

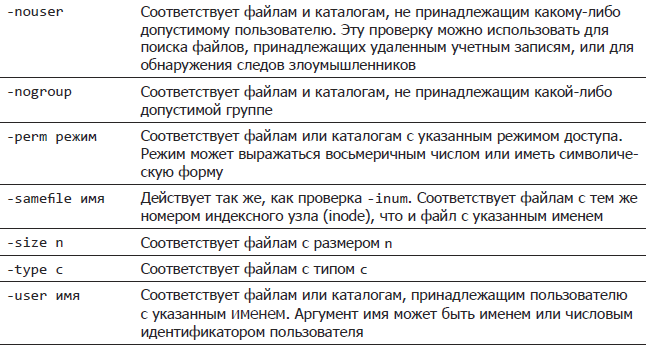
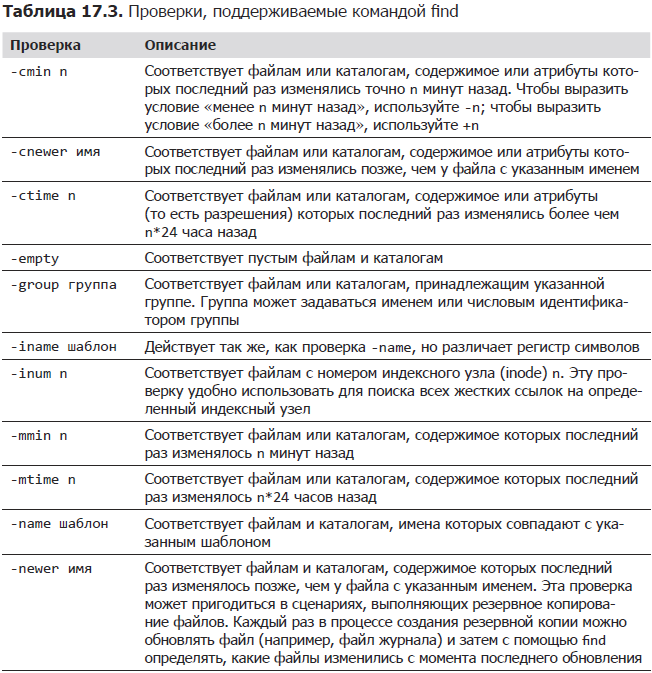


Добавив дополнительные проверки, можно выполнять поиск файлов по размеру и имени. Давайте найдем все обычные файлы с именами, соответствующими ша­блону *\*.JPG*, и имеющие размер больше 1 мегабайта:

[me@linuxbox ~]$ **find ~ -type f -name "\*.JPG" -size +1M | wc –l**

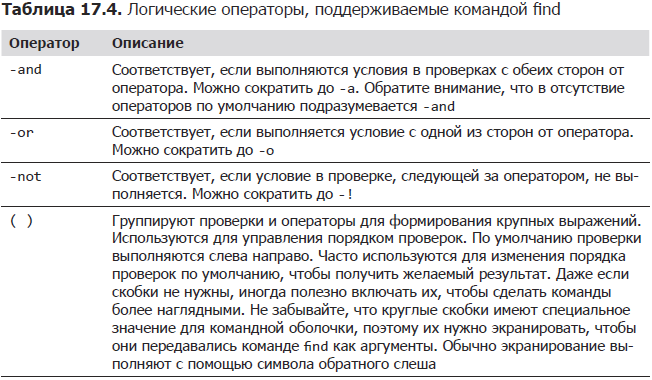


Команда find поддерживает множество разнообразных проверок. В табл. 17.3 при­водится краткое описание наиболее часто используемых из них.



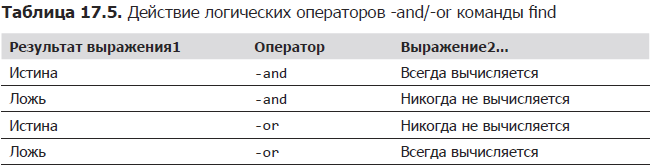
На­пример, представьте, что в некотором каталоге мы хотим найти все файлы и под­каталоги с небезопасными разрешениями. Для этого можно было бы выполнить поиск всех файлов с разрешениями, отличающимися от 0600, и каталогов с раз­решениями, отличающимися от 0700. К счастью, find поддерживает возможность комбинирования проверок с помощью логических операторов, с целью опреде­лить более сложные критерии отбора.

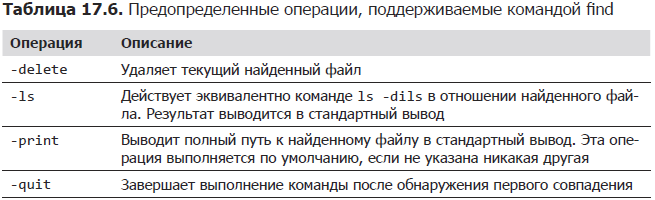
[me@linuxbox ~]$ **find ~ \( -type f -not -perm 0600 \) -or \( -type d -not –perm 0700 \)**



*выражение1* -оператор *выражение2*

*Выражение1* будет вычислено в любом случае, а вот будет ли вычислено *выраже­ние2*, зависит от оператора. В табл. 17.5 показано, как это работает.





[me@linuxbox ~]$ **find ~ \( -ls -type f -not -perm 0600 \) -or \( -ls -type d -not –perm 0700 \)**

Программу find можно использовать для удаления файлов, соответствующих определенным критериям.

**find ~ -type f -name '\*.BAK' -delete**

Обратите внимание, что операцию -delete следует использовать **с особыми предосто­рожностями**. Всегда предварительно проверяйте команду, подставив операцию -print вместо –delete

**find ~ -print -and -type f -and -name '\*.BAK'**

Эта версия команды выведет каждый файл (операция -print всегда возвращает истинное значение), а затем проверит тип файла и его расширение.

**Операции, определяемые пользователем**

Помимо предопределенных операций можно также вызывать произвольные команды.

-exec *команда* {} ;

где команда — это имя команды, {} — символическое представление текущего пути к файлу и точка с запятой — обязательный разделитель, обозначающий конец команды.

Следующий пример демонстрирует использование -exec для получения эффекта, аналогичного операции -delete, обсуждавшейся выше:

-exec rm '{}' ';'

**find ~ -type f -name 'foo\*' -ok ls -l '{}' ';'**

< ls ... /home/me/bin/foo > ? **y**

-rwxr-xr-x 1 me me 224 2011-10-29 18:44 /home/me/bin/foo

Если заменить завершающий символ точки с запятой знаком «плюс», в команде find активируется функция объединения результатов в список аргументов для вы­зова единственного экземпляра требуемой команды. Вернемся к нашему примеру. Команда:

**find ~ -type f -name 'foo\*' -exec ls -l '{}' ';'**

-rwxr-xr-x 1 me me 224 2011-10-29 18:44 /home/me/bin/foo

-rw-r--r-- 1 me me 0 2012-09-19 12:53 /home/me/foo.txt

будет вызывать ls для каждого найденного файла. Изменив команду, как показа­но ниже:

**find ~ -type f -name 'foo\*' -exec ls -l '{}' +**

-rwxr-xr-x 1 me me 224 2011-10-29 18:44 /home/me/bin/foo

-rw-r--r-- 1 me me 0 2012-09-19 12:53 /home/me/foo.txt

мы получим тот же результат, но система выполнит команду ls только один раз.

Тот же результат можно получить с помощью команды xargs. Она принимает входные данные со стандартного ввода и преобразует их в список аргументов для указанной команды. В данном примере ее можно было бы использовать так:

**find ~ -type f -name 'foo\*' -print | xargs ls -l**

-rwxr-xr-x 1 me me 224 2011-10-29 18:44 /home/me/bin/foo

-rw-r--r-- 1 me me 0 2012-09-19 12:53 /home/me/foo.txt

Команда find поддерживает операцию -print0, которая производит вывод имен файлов, разделенных пустым символом, а команда xargs имеет параметр --null, позволяющий организовать прием значений, разделенных пустым символом. Например:

find ~ -iname '\*.jpg' -print0 | xargs --null ls -l

Этот прием гарантирует правильную обработку любых имен файлов, даже содержащих пробелы.

Сначала создадим песочницу с множеством файлов и каталогов:

[me@linuxbox ~]$ **mkdir -p playground/dir-{00{1..9},0{10..99},100}**

[me@linuxbox ~]$ **touch playground/dir-{00{1..9},0{10..99},100}/file-{A..Z}**

Какая мощь командной строки! Эти две строки создают каталог playground, со­держащий 100 подкаталогов и 26 пустых файлов в каждом.

Эта команда создаст пустой файл *timestamp* и установит время его последнего из­менения равным текущему времени.

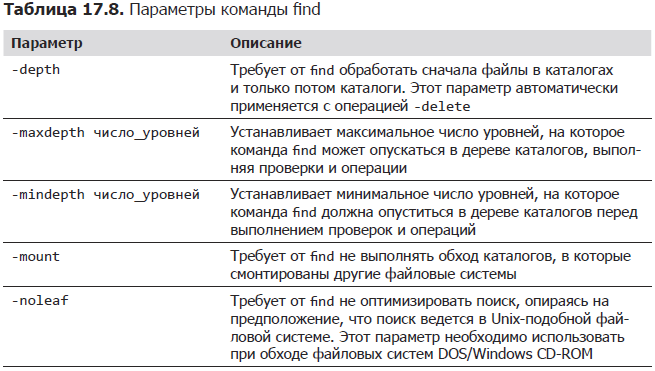
[me@linuxbox ~]$ **touch playground/timestamp**

Мы можем убедиться в этом, использовав еще одну полезную команду, stat, которую можно рассматривать как своего рода форсированную версию ls. Команда stat выводит всю информацию о файле

[me@linuxbox ~]$ **stat playground/timestamp**

Теперь найдем с помощью find обновленные фай­лы, сравнив все файлы с эталонным файлом *timestamp*:

[me@linuxbox ~]$ **find playground -type f -newer playground/timestamp**

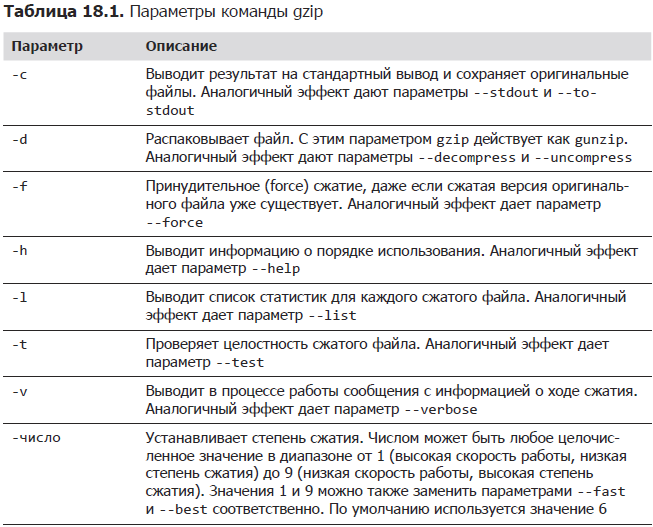


----------------------------------------------------------

**gzip — сжатие и распаковывание файлов**

[me@linuxbox ~]$ **gzip foo.txt**

[me@linuxbox ~]$ **gunzip foo.txt**



**bzip2 — высокая степень сжатия ценой скорости**

[me@linuxbox ~]$ **bzip2 foo.txt**

[me@linuxbox ~]$ **bunzip2 foo.txt.bz2**

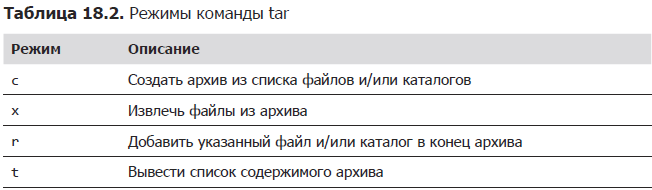
Как видите, bzip2 можно использовать так же, как gzip. Все параметры програм­мы gzip (кроме -r), представленные выше, поддерживаются также программой bzip2.

**tar — утилита архивирования**

tar *режим*[*параметры*] *путь*...

[me@linuxbox ~]$ **tar cf playground.tar playground**

Эта команда создаст tar-архив с именем *playground.tar*, включающий всю иерар­хию каталогов песочницы. Как видите, режим и параметр f, который использует­ся для определения имени tar-архива, можно объединять, и при этом не требуется использовать начальный дефис. Но имейте в виду, что режим всегда должен ука­зываться первым, перед любыми параметрами.



Посмотреть содержимое архива можно с помощью следующей команды:

[me@linuxbox ~]$ **tar tf playground.tar**

Для получения более подробного списка добавим параметр v (verbose — подроб­ности):

[me@linuxbox ~]$ **tar tvf playground.tar**

Теперь извлечем содержимое архива в другой каталог. Для этого создадим новый каталог с именем *foo*, перейдем в него и извлечем содержимое tar-архива:

[me@linuxbox ~]$ **mkdir foo**

[me@linuxbox ~]$ **cd foo**

[me@linuxbox foo]$ **tar xf ../playground.tar**

Однако следует помнить, что если вы не действуете от имени суперпользо­вателя, файлы и каталоги, извлеченные из архива, будут принадлежать пользовате­лю, выполнившему восстановление, а не первоначальному их владельцу.

**zip — упаковывание и сжатие файлов**

Программа zip одновременно является и инструментом сжатия, Однако в Linux чаще дру­гих используется программа сжатия gzip, а второе место занимает bzip2. Поль­зователи Linux используют zip в основном для обмена файлами с системами Windows.

[me@linuxbox ~]$ **zip -r playground.zip playground**

Без параметра -r (отвечает за рекурсивный обход каталогов) в архив будет вклю­чен только каталог *playground* (без своего содержимого).

[me@linuxbox foo]$ **unzip ../playground.zip**

С помо­щью параметра -@ программе zip по конвейеру передается список имен файлов:

[me@linuxbox ~]$ **find playground -name "file-A" | zip -@ file-A.zip**

Здесь команда find генерирует список файлов, соответствующих проверке -name "file-A", и передает его по конвейеру команде zip, которая затем создает архив *file-A.zip* с выбранными файлами.

**rsync — синхронизация файлов и каталогов с удаленной системой**

Эта программа синхронизует локальные и удаленные каталоги, используя протокол *rsync remote-update,* который позволяет rsync быстро обнаруживать различия между двумя каталога­ми и копировать минимальный объем данных.

rsync *параметры источник приемник*

где роль источника и приемника могут играть:

1. локальный файл или каталог;
2. удаленный файл или каталог в форме *[пользователь@]хост:путь*;
3. удаленный сервер rsync, определяемый идентификатором URI *rsync://[поль­зователь@]хост[:порт]/путь*.

Обратите внимание, что либо источник, либо приемник должен находиться в ло­кальной системе. Копирование из удаленной системы в удаленную систему не поддерживается.

Далее синхронизируем каталог *playground* с соответствующей копией в *foo*:

[me@linuxbox ~]$ **rsync -av playground foo**

Мы добавили два параметра: -a (для архивирования — обеспечивает рекурсивный обход и сохранение атрибутов файлов) и -v (подробный вывод)

Если после подключения такого диска к системе он снова будет смонтирован в каталог */media/BigDisk*, выполним первое резервное копирование системы, для начала создав каталог */backup* на внешнем устройстве, а затем вызвав rsync для копирования наиболее важных компонентов.

[me@linuxbox ~]$ **mkdir /media/BigDisk/backup**

[me@linuxbox ~]$ **sudo rsync -av --delete /etc /home /usr/local /media/BigDisk/backup**

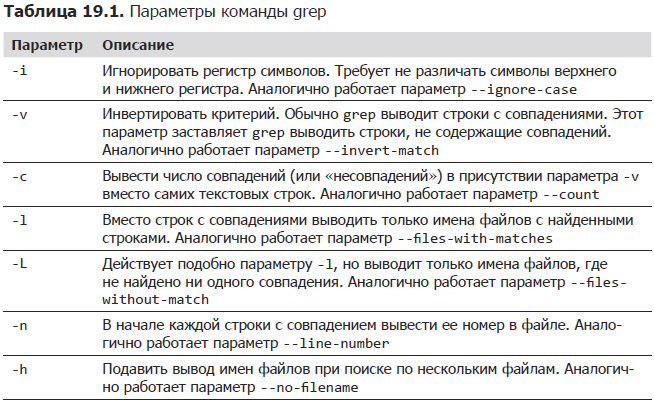
В этом примере мы скопировали каталоги */etc*, */home* и */usr/local* из нашей си­стемы на воображаемый внешний диск. Мы добавили параметр --delete, чтобы удалить файлы, которые могут присутствовать на устройстве с резервной копией, но отсутствовать на устройстве-источнике (этот параметр не нужен при создании резервной копии в первый раз, но является полезным дополнением в последую­щих операциях копирования).

-----------------------------------------------------

**grep — поиск в тексте**

Программа grep имеет следующий синтаксис:

grep [*параметры*] *регулярное\_выражение* [*файл*...]



Ниже показано, как выполнить простой поиск в нашем списке файлов:

[me@linuxbox ~]$ **grep bzip dirlist\*.txt**

В этом примере grep просматривает все перечисленные файлы в поисках строки bzip

Если бы нам достаточно было получить только имена файлов с совпадениями, а не сами совпадения, мы могли бы добавить параметр -l:

[me@linuxbox ~]$ **grep -l bzip dirlist\*.txt**

Напротив, получить список файлов, не содержащих совпадений, можно так:

[me@linuxbox ~]$ **grep -L bzip dirlist\*.txt**

**Метасимволы и литералы**

К метасимволам регулярных выражений относятся следующие символы:

^ $ . [ ] { } - ? \* + ( ) | \

**Любой символ**

Символ точки, соответствую­щий любому символу.

[me@linuxbox ~]$ **grep -h '.zip' dirlist\*.txt**

bunzip2

bzip2

**Якоря**

Символ крышки (^) и знак доллара ($) в регулярных выражениях интерпрети­руются как *якоря*. Это означает, что в их присутствии совпадение с регулярным выражением возможно, только если совпадение будет найдено в начале строки (^) или в ее конце ($).

[me@linuxbox ~]$ **grep -h '^zip' dirlist\*.txt**

zip

zipcloak

[me@linuxbox ~]$ **grep -h 'zip$' dirlist\*.txt**

gunzip

gzip

[me@linuxbox ~]$ **grep -h '^zip$' dirlist\*.txt**

zip

**Выражения в квадратных скобках и классы символов**

В следующем примере используется множе­ство из двух символов, благодаря которому обнаруживаются соответствия с по­следовательностями bzip и gzip:

[me@linuxbox ~]$ **grep -h '[bg]zip' dirlist\*.txt**

bzip2

bzip2recover

gzip

Пер­вый — символ крышки (^), который используется для обозначения отрицания; второй — дефис (-), который используется для обозначения диапазона символов.

**Отрицание**

Если сразу после открывающей квадратной скобки стоит символ крышки (^), остальные символы множества интерпретируются как недопустимые в данной позиции.

[me@linuxbox ~]$ **grep -h '[^bg]zip' dirlist\*.txt**

bunzip2

gunzip

**Традиционные диапазоны символов**

Если необходимо сконструировать регулярное выражение, которое находило бы в наших списках все файлы с именами, начинающимися с заглавной буквы

[me@linuxbox ~]$ **grep -h '^[ABCDEFGHIJKLMNOPQRSTUVWXZY]' dirlist\*.txt**

Но необходимость ввода всех этих символов вызывает некоторое беспокойство, поэтому предусмотрен другой способ:

[me@linuxbox ~]$ **grep -h '^[A-Z]' dirlist\*.txt**

MAKEDEV

ControlPanel

Например, для поиска имен файлов, начинающихся с буквы или цифры:

[me@linuxbox ~]$ **grep -h '^[A-Za-z0-9]' dirlist\*.txt**

С другой стороны, следующее выражение:

[me@linuxbox ~]$ **grep -h '[-AZ]' dirlist\*.txt**

найдет все имена файлов, содержащие дефис, букву *A* или букву *Z*.

**Классы символов POSIX**

[me@linuxbox ~]$ **ls /usr/sbin/[ABCDEFGHIJKLMNOPQRSTUVWXYZ]\***

/usr/sbin/MAKEFLOPPIES

/usr/sbin/NetworkManagerDispatcher

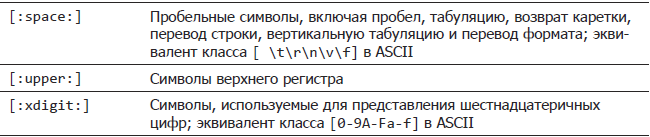
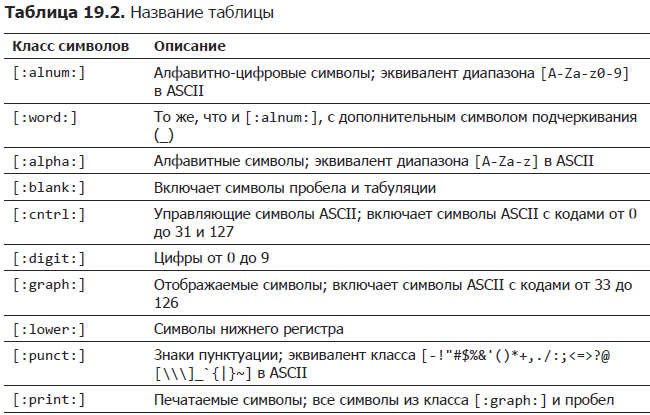
Эта команда вернула ожидаемый результат — список имен файлов, начинающихся с заглавной буквы. Но следующая команда даст совершенно другой результат (здесь приведена толь­ко часть результатов):

[me@linuxbox ~]$ **ls /usr/sbin/[A-Z]\***

/usr/sbin/biosdecode

/usr/sbin/chat

В чем же причина?



[me@linuxbox ~]$ **ls /usr/sbin/[[:upper:]]\***

/usr/sbin/MAKEFLOPPIES

/usr/sbin/NetworkManagerDispatcher

**Простые и расширенные регулярные выражения POSIX**

Стандарт POSIX также делит реализации ре­гулярных выражений на два вида: *простые регулярные выражения* (Basic Regular Expressions, BRE) и *расширенные регулярные выражения* (Extended Regular Expressions, ERE).

**Чередование**

Первой особенностью расширенных регулярных выражений, которую мы об­судим, будет *чередование*

Сначала по­пробуем выполнить простое сопоставление строк:

[me@linuxbox ~]$ **echo "AAA" | grep AAA**

AAA

Теперь добавим чередование, обозначаемое метасимволом вертикальной черты:

[me@linuxbox ~]$ **echo "AAA" | grep -E 'AAA|BBB'**

AAA

Мы до­бавили в команду grep параметр -E (вместо этого можно было бы использовать программу egrep) и заключили регулярное выражение в кавычки, чтобы предот­вратить интерпретацию командной оболочкой символа вертикальной черты как оператора конвейера.

Для объединения с другими элементами регулярного выражения чередование можно заключать в круглые скобки ():

[me@linuxbox ~]$ **grep -Eh '^(bz|gz|zip)' dirlist\*.txt**

Этому выражению будут соответствовать имена файлов из наших списков, начи­нающиеся с bz, gz или zip.

**? — совпадение с элементом ноль или один раз**

Этот квантификатор фактически означает: «совпадение с предыдущим элементом не обязательно».

[me@linuxbox ~]$ **echo "(555) 123-4567" | grep -E '^\(?[0-9][0-9][0-9]\)? [0-9][0-9][0-9]$'**

(555) 123-4567

В этом выражении за круглыми скобками следуют знаки вопроса, указывающие, что скобки могут либо отсутствовать, либо присутствовать один раз.

**\* — совпадение с элементом ноль или более раз**

Подобно метасимволу ?, звездочка (\*) обозначает необязательный элемент; однако, в отличие от знака вопроса (?), этот элемент может встречаться любое число раз, а не только единожды.

[me@linuxbox ~]$ **echo "This works." | grep -E '[[:upper:]][[:upper:][:lower:] ]\*\.'**

This works.

**+ — совпадение с элементом один или более раз**

Метасимвол + действует почти так же, как \*, но требует совпадения с предыдущим элементом не менее одного раза.

[me@linuxbox ~]$ **echo "This that" | grep -E '^([[:alpha:]]+ ?)+$'**

This that

[me@linuxbox ~]$ **echo "a b c" | grep -E '^([[:alpha:]]+ ?)+$'**

a b c

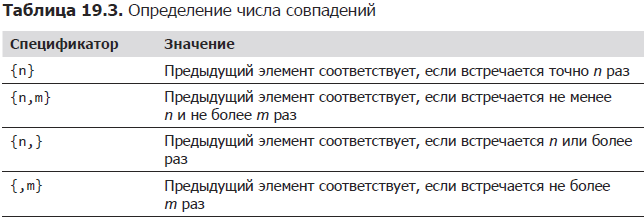
[me@linuxbox ~]$ **echo "a b 9" | grep -E '^([[:alpha:]]+ ?)+$'**

[me@linuxbox ~]$ **echo "abc d" | grep -E '^([[:alpha:]]+ ?)+$'**

Как видите, этому выражению не соответствует строка "a b 9", потому что она содержит неалфавитный символ; точно так же ему не соответствует строка "abc d", потому что между символами c и d в ней присутствует больше одного пробела.

**{ } — совпадение с элементом определенное число раз**

Метасимволы { и } используются, чтобы выразить минимальное и максимальное число обязательных совпадений.



[me@linuxbox ~]$ **echo "(555) 123-4567" | grep -E '^\(?[0-9]{3}\)? [0-9]{3}-[0-9]{4}$'**

(555) 123-4567

**Поиск необычных имен файлов с помощью find**

В результате такого поиска можно выявить имена файлов и каталогов, содержа­щие пробелы и другие, потенциально вредные символы:

[me@linuxbox ~]$ **find . -regex '.\*[^-\_./0-9a-zA-Z].\*'**

[me@linuxbox ~]$ find /bin/ -regex **'.\*bz.\*'**

/bin/bzcat

/bin/bzcmp

/bin/bzdiff

Из-за требования точного совпадения всего пути мы добавили элемент .\* с обоих концов выражения, замещающий любое количество любых символов (в том чис­ле и отсутствие символов).

**Поиск файлов с помощью locate**

[me@linuxbox ~]$ **locate --regex '/bin/(bz|gz|zip)'**

/bin/bzcat

/bin/bzcmp

/bin/bzdiff

/bin/gzip

/usr/bin/zip

**Поиск текста в less и vim**

less и vim поддерживают одинаковые способы поиска в тексте. Чтобы выполнить поиск, нажмите клавишу / и введите регулярное выражение.

[me@linuxbox ~]$ **less phonelist.txt**

(458) 273-1642

(686) 299-8268

(198) 307-2440

~

~

~

/^\([0-9]{3}\) [0-9]{3}-[0-9]{4}$

less выделит строки с совпадениями, что позволит сразу увидеть недопустимые номера

Редактор vim поддерживает только простые регулярные выражения, поэтому вы­ражение для поиска должно выглядеть следующим образом:

/([0-9]\{3\}) [0-9]\{3\}-[0-9]\{4\}

-------------------------------------------------

**ТЕКСТ В MS-DOS И UNIX**

**cat — объединение файлов и вывод их в стандартный поток вывода**

Существует несколько способов преобразовать файлы из формата DOS в формат Unix. Во многих системах Linux имеются программы unix2dos и dos2unix для преоб­разования текстовых файлов в формат DOS и обратно.

Программа cat содержит множество интересных параметров. Примером может служить параметр -A, используемый для отображения непечатаемых сим­волов в тексте.

Для этого введем команду cat (указав файл для перенаправ­ления вывода), а следом наш текст, завершив строки нажатием клавиши ENTER и закончив все комбинацией CTRL+D — она сообщит программе cat, что достиг­нут конец файла.

[me@linuxbox ~]$ **cat > foo.txt**

**The quick brown fox jumped over the lazy dog**1**.**

[me@linuxbox ~]$

Далее, вызовем cat с параметром -A, чтобы показать текст:

[me@linuxbox ~]$ **cat -A foo.txt**

^IThe quick brown fox jumped over the lazy dog. $

Как видите, символ табуляции в тексте представлен парой символов ^I. Эта обыч­ная форма записи означает «CTRL+I», то есть, как оказывается, — символ табуля­ции. Здесь также видно, что символ $ отмечает истинный конец строки, помогая увидеть дополнительные пробелы в конце строки.

Программа cat имеет также параметры, используемые для изменения текста. Наи­более известными являются -n, добавляющий номера строк, и -s, подавляющий вывод множества пустых строк, идущих подряд.

[me@linuxbox ~]$ **cat > foo.txt**

**The quick brown fox**

**jumped over the lazy dog.**

[me@linuxbox ~]$ **cat -ns foo.txt**

1 The quick brown fox

2

3 jumped over the lazy dog.

[me@linuxbox ~]$

После обработки текста командой cat с параметрами -ns одна пустая строка была удалена, а остальные строки пронумерованы.

**sort — сортировка строк текстовых файлов**

Программа sort сортирует содержимое стандартного ввода или одного или не­скольких файлов, указанных в командной строке, и записывает результаты в стан­дартный вывод.

[me@linuxbox ~]$ **sort > foo.txt**

c

b

a

[me@linuxbox ~]$ **cat foo.txt**

a

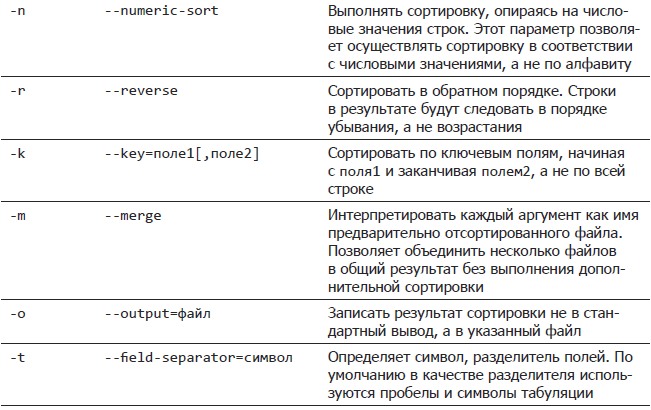
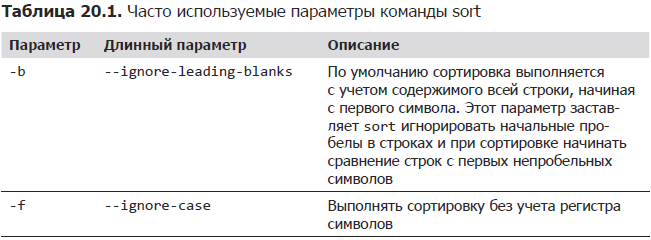
b

c

После запуска команды мы ввели буквы c, b и a, а затем признак конца файла с по­мощью комбинации CTRL+D. Затем просмотрели получившийся файл и увидели, что строки в нем отсортированы.

Поскольку sort может принимать несколько файлов в аргументах командной строки, существует возможность объединить множество файлов в один отсорти­рованный файл.

[me@linuxbox ~]$ sort file1.txt file2.txt file3.txt > final\_sorted\_list.txt



Du

Обычно команда du выводит результаты, отсортированные по име­нам каталогов:

[me@linuxbox ~]$ **du -s /usr/share/\* | head**

252 /usr/share/aclocal

96 /usr/share/acpi-support

8 /usr/share/adduser

196 /usr/share/alacarte

344 /usr/share/alsa

8 /usr/share/alsa-base

[me@linuxbox ~]$ **du -s /usr/share/\* | sort -nr | head**

509940 /usr/share/locale-langpack

242660 /usr/share/doc

197560 /usr/share/fonts

С помощью параметров -nr мы получили сортировку по числовым значениям в обратном порядке

ls может сортировать свои результаты по размеру, и выпол­ним сортировку с помощью sort.

[me@linuxbox ~]$ **ls -l /usr/bin | sort -nr -k 5 | head**

-rwxr-xr-x 1 root root 8234216 2012-04-07 17:42 inkscape

-rwxr-xr-x 1 root root 8222692 2012-04-07 17:42 inkview

-rwxr-xr-x 1 root root 3746508 2012-03-07 23:45 gimp-2.4

-rwxr-xr-x 1 root root 3654020 2012-08-26 16:16 quanta

параметр -k с ар­гументом 5, чтобы сортировка выполнялась по пятому полю.

Если диапазон не определен, sort использует в качестве ключа часть строки, начинающуюся с указанного поля и простирающуюся до конца строки.

[me@linuxbox ~]$ **sort --key=1,1 --key=2n distros.txt**

Fedora 5 03/20/2006

Fedora 6 10/24/2006

Fedora 7 05/31/2007

Fedora 8 11/08/2007

Fedora 9 05/13/2008

Fedora 10 11/25/2008

SUSE 10.1 05/11/2006

С тем же успехом можно было бы передать параметры -k 1,1 -k 2n.

мы указали диапазон 1,1, что означает: «начиная с поля 1 и заканчивая полем 1». Аргумент 2n, который означает: «ключом сорти­ровки является второе поле и сортировка выполняется в порядке числовых зна­чений».

Параметр --key позволяет определять смещения внутри полей, чтобы в качестве ключей можно было ис­пользовать части полей:

[me@linuxbox ~]$ **sort -k 3.7nbr -k 3.1nbr -k 3.4nbr distros.txt**

Fedora 10 11/25/2008

Ubuntu 8.10 10/30/2008

SUSE 11.0 06/19/2008

Fedora 9 05/13/2008

Ubuntu 8.04 04/24/2008

Fedora 8 11/08/2007

Добавив параметр -k 3.7, мы сообщили программе sort, что она должна исполь­зовать для сортировки ключ, начинающийся с седьмого символа в третьем поле. Мы также добавили параме­тры n и r, чтобы выполнить числовую сортировку в обратном порядке. Параметр b добавлен для исключения начальных пробелов.

Про­грамма sort поддерживает параметр -t для определения символа-разделителя по­лей. Чтобы отсортировать содержимое файла *passwd* по седьмому полю, используем такую команду:

[me@linuxbox ~]$ **sort -t ':' -k 7 /etc/passwd | head**

me:x:1001:1001:Myself,,,:/home/me:/bin/bash

root:x:0:0:root:/root:/bin/bash

dhcp:x:101:102::/nonexistent:/bin/false

gdm:x:106:114:Gnome Display Manager:/var/lib/gdm:/bin/false

Определив двоеточие как разделитель полей.

sort поддерживает параметр -u, удаляющий повто­ряющиеся строки из сортированных результатов.

**uniq — выявление или удаление повторяющихся строк**

Она удаляет повторяющиеся строки и выводит результат в стандартный вывод.

[me@linuxbox ~]$ **cat > foo.txt**

**a**

**b**

**c**

**a**

**b**

**c**

Не забудьте ввести CTRL+D, чтобы завершить ввод с клавиатуры.

Чтобы uniq действительно выполнила свою работу, исходные данные нужно сна­чала отсортировать:

[me@linuxbox ~]$ **sort foo.txt | uniq**

a

b

c

uniq удаляет повторяющиеся записи, только если они следуют друг за другом.

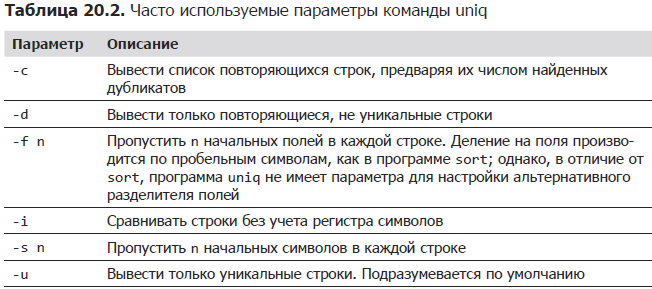
параметр -c программы uniq для определе­ния числа повторяющихся строк в исходном текстовом файле:

[me@linuxbox ~]$ **sort foo.txt | uniq -c**

2 a

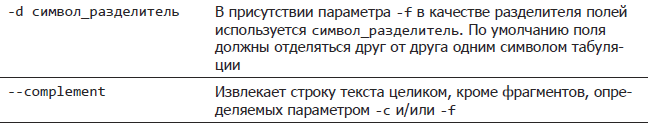
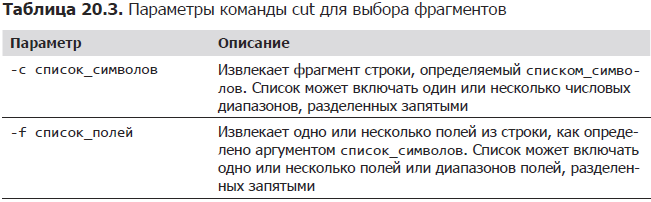
2 b

2 c



**cut — удаление фрагментов из всех строк в файлах**

Программа cut используется для извлечения фрагментов текста из строк и выво­да их в стандартный вывод.



[me@linuxbox ~]$ **cat -A distros.txt**

SUSE^I10.2^I12/07/2006$

Fedora^I10^I11/25/2008$

SUSE^I11.0^I06/19/2008$

Поскольку вместо пробелов в файле используются символы табуляции, можно воспользоваться параметром -f для извлечения поля:

[me@linuxbox ~]$ **cut -f 3 distros.txt**

12/07/2006

11/25/2008

06/19/2008

04/24/2008

[me@linuxbox ~]$ **cut -f 3 distros.txt | cut -c 7-10**

2006

2008

2008

2008

Применив cut второй раз к нашему списку, мы смогли извлечь символы в позициях с 7-й по 10-ю

[me@linuxbox ~]$ **cut -d ':' -f 1 /etc/passwd | head**

root

daemon

bin

sys

sync

С помощью параметра -d мы определили, что роль разделителя полей будет играть символ двоеточия.

**paste — слияние строк из файлов**

С помощью paste добавим колонку с датами перед названиями и номерами версий дистрибутивов, создав хронологический список. Для этого достаточно просто вызвать paste и пе­редать ей файлы в требуемом порядке.

[me@linuxbox ~]$ **paste distros-dates.txt distros-versions.txt**

11/25/2008 Fedora 10

10/30/2008 Ubuntu 8.10

06/19/2008 SUSE 11.0

05/13/2008 Fedora 9

04/24/2008 Ubuntu 8.04

11/08/2007 Fedora 8

10/18/2007 Ubuntu 7.10

**join — объединение строк из двух файлов по общему полю**

Книга -- Командная строка Linux. Полное руководство стр.281

**comm — построчное сравнение двух сортированных файлов**

Программа comm сравнивает два текстовых файла, показывая, какие строки в них уникальные, а какие — одинаковые. Для демонстрации создадим с помощью cat два почти идентичных файла:

[me@linuxbox ~]$ **cat > file1.txt**

**a**

**b**

**c**

**d**

[me@linuxbox ~]$ **cat > file2.txt**

**b**

**c**

**d**

**e**

Затем сравним эти два файла с помощью comm:

[me@linuxbox ~]$ **comm file1.txt file2.txt**

a

b

c

d

e

Как видите, comm произвела вывод в три колонки. Первая колонка содержит уни­кальные строки из первого файла, вторая — уникальные строки из второго фай­ла, третья — строки, одинаковые в обоих файлах. Программа comm поддерживает параметры в формате -n, где n может быть числом 1, 2 или 3. При использовании эти параметры определяют номера колонок, вывод которых следует подавить. Например, чтобы вывести только одинаковые строки, нужно подавить вывод колонок 1 и 2:

[me@linuxbox ~]$ **comm -12 file1.txt file2.txt**

b

c

d

**diff — построчное сравнение файлов**

Подобно программе comm, diff используется для выявления различий между фай­лами. Программа diff часто используется разработчиками программного обе­спечения для исследования различий между разными версиями исходного про­граммного кода, потому что позволяет рекурсивно обходить каталоги, которые часто называют *деревьями исходного кода*

Если применить diff к файлам из предыдущего примера, можно увидеть стиль вывода результатов ее работы по умолчанию: краткое описание различий между двумя файлами.

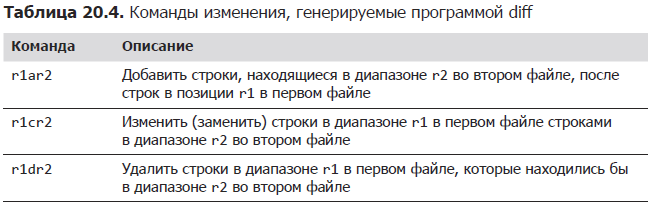
[me@linuxbox ~]$ **diff file1.txt file2.txt**

1d0

< a

4a4

> e



Хотя этот формат используется по умолчанию, он не так широко исполь­зуется, как другие, дополнительные форматы. Два других формата, получив­ших большую популярность, — это *контекстный формат* и *унифицированный формат*.

При использовании контекстного формата (параметр -c) вывод выглядит так:

[me@linuxbox ~]$ **diff -c file1.txt file2.txt**

\*\*\* file1.txt 2012-12-23 06:40:13.000000000 -0500

--- file2.txt 2012-12-23 06:40:34.000000000 -0500

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* 1,4 \*\*\*\*

- a

b

c

d

--- 1,4 ----

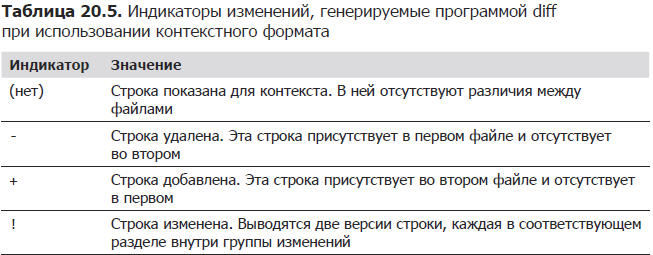
b

c

d

+ e

Вывод начинается с имен двух файлов и времени последнего их изменения. Пер­вый файл отмечается звездочками, а второй — дефисами. Первая группа начинается со строки \*\*\* 1,4 \*\*\*\*, указывающей на строки с номерами с 1 по 4 в первом файле. Далее следует строка --- 1,4 ----, указывающая на строки с номерами с 1-го по 4-й во втором файле. Внутри группы изменений присутствуют строки, начинающиеся с одного из четырех индикаторов, перечисленных в табл. 20.5.



Унифицированный формат напоминает контекстный, но более компактный. За­дается параметром -u:

[me@linuxbox ~]$ **diff -u file1.txt file2.txt**

--- file1.txt 2012-12-23 06:40:13.000000000 -0500

+++ file2.txt 2012-12-23 06:40:34.000000000 -0500

@@ -1,4 +1,4 @@

-a

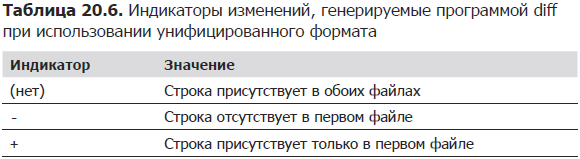
b

c

d

+e

В примере, приведенном выше, видны те же времена последнего изменения фай­лов, что и в контекстном формате, за которыми следует строка @@ -1,4 +1,4 @@. Она указывает номера строк в первом и во втором файлах, описываемых группой изменений. Далее следуют сами строки с тремя (по умолчанию) строками контек­ста



**patch — применение diff-файла к оригиналу**

Программа patch используется для применения изменений к текстовым файлам. Она принимает вывод программы diff и обычно используется для преобразования старых версий файлов в более новые.

Чтобы подготовить diff-файл для последующего его применения программой patch, документация GNU предлагает использовать diff, как показано ниже:

diff -Naur *старый\_файл новый\_файл* > *diff\_файл*

где старый\_файл и новый\_файл могут быть одиночными файлами или каталогами. Параметр r поддерживает рекурсивный обход вложенных подкаталогов.

[me@linuxbox ~]$ **diff -Naur file1.txt file2.txt > patchfile.txt**

[me@linuxbox ~]$ **patch < patchfile.txt**

patching file file1.txt

[me@linuxbox ~]$ **cat file1.txt**

b

c

d

e

После наложения «заплаты» содержимое *file1.txt* точно соответствует содержимому *file2.txt*.

**tr — перекодирование или удаление символов**

Программа tr используется для *перекодирования* символов. Ее можно рассма­тривать как своеобразную посимвольную операцию поиска с заменой.

[me@linuxbox ~]$ **echo "lowercase letters" | tr a-z A-Z**

LOWERCASE LETTERS

Множества символов можно выразить тремя способами:

1. в виде списка-перечисления, например ABCDEFGHIJKLMNOPQRSTUVWXYZ;
2. в виде диапазона символов, например A-Z. Обратите внимание, что этому спо­собу сопутствуют те же проблемы, что наблюдаются в других программах (из-за разного порядка алфавитной сортировки в разных регионах), и потому он должен использоваться с осторожностью;
3. в виде классов символов POSIX, например [:upper:].

[me@linuxbox ~]$ **echo "lowercase letters" |** tr '[:lower:]' '[:upper:]'

LOWERCASE LETTERS

Выше в этой главе мы обсуждали проблему преобразования текстовых файлов в формате MS-DOS в текст в формате Unix. Для такого преобразования достаточ­но просто удалить символы возврата каретки в конце каждой строки. Эту опера­цию можно выполнить с помощью tr:

tr -d '\r' < *файл\_dos* > *файл\_unix*

где файл\_dos — это файл, подлежащий преобразованию, а файл\_unix — результат.

**ROT13: НЕ САМЫЙ НАДЕЖНЫЙ СПОСОБ ШИФРОВАНИЯ**

Одно забавное применение команды tr — *шифрование текста по алгоритму ROT13*. ROT13 — тривиальный тип шифрования, основанный на простом подстановочном шиф­ре. Метод заключается в простом смещении каждого символа на 13 позиций далее по алфавиту. Так как число 13 соответствует середине набора из 26 символов, повторное применение алгоритма к тексту приводит к его восстановлению в исходное состояние.

echo "secret text" | tr a-zA-Z n-za-mN-ZA-M

frperg grkg

Повторное применение той же процедуры приводит к обратному преобразованию:

echo "frperg grkg" | tr a-zA-Z n-za-mN-ZA-M

secret text

При вызове с параметром -s команда tr «сжимает» (squeeze), или удаляет повторяющиеся экземпляры символов:

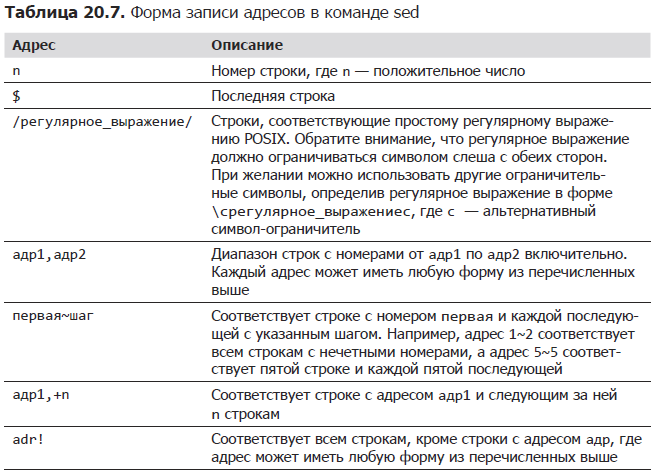
[me@linuxbox ~]$ **echo "aaabbbccc" | tr -s ab**

Abccc

Обратите внимание, что повторяющиеся символы должны следо­вать подряд. В противном случае сжатие не даст никакого эффекта

**sed — потоковый редактор для фильтрации и преобразования текста**

Имя sed — это сокращенное словосочетание *stream editor* (потоковый редактор). Данная команда осуществляет редактирование потока текста, получаемого из множества файлов или подаваемого на стандартный ввод команды.



[me@linuxbox ~]$ **sed -n '1,5p' distros.txt**

SUSE 10.2 12/07/2006

Fedora 10 11/25/2008

SUSE 11.0 06/19/2008

Ubuntu 8.04 04/24/2008

Fedora 8 11/08/2007

В нашем примере мы вывели строки с 1 по 5. Для этого использовалась коман­да p, которая просто выводит строки, соответствующие адресам. Однако здесь нам пришлось добавить параметр -n (параметр подавления автоматического вывода), чтобы программа sed не выводила все строки.

[me@linuxbox ~]$ **sed -n '/SUSE/p' distros.txt**

SUSE 10.2 12/07/2006

SUSE 11.0 06/19/2008

SUSE 10.3 10/04/2007

SUSE 10.1 05/11/2006

Включив регулярное выражение /SUSE/, заключенное в символы слеша, мы смог­ли выделить строки подобно тому, как это делает программа grep.

Наконец, попробуем применить оператор отрицания, добавив в адрес восклица­тельный знак (!):

[me@linuxbox ~]$ **sed -n '/SUSE/!p' distros.txt**

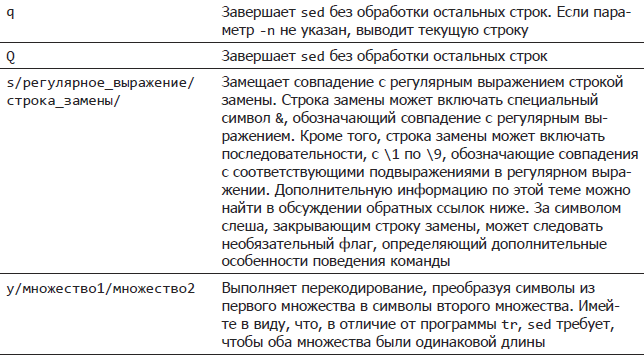
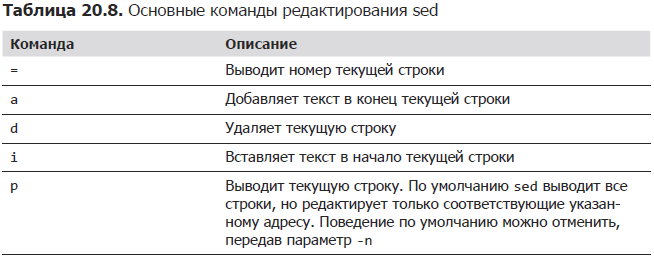
Fedora 10 11/25/2008

Ubuntu 8.04 04/24/2008

Fedora 8 11/08/2007

Ubuntu 6.10 10/26/2006

Здесь мы видим ожидаемый результат: все строки из файла, кроме совпавших с регулярным выражением.



Перевод даты в формате ММ/ДД/ГГГГ, в формат ГГГГ-ММ-ДД. Замена пред­ставления дат вручную — довольно утомительное занятие и чревато ошибками, но с помощью sed ту же замену можно выполнить в одно действие:

[me@linuxbox ~]$ **sed 's/\([0-9]\{2\}\)\/\([0-9]\{2\}\)\/\([0-9]\{4\}\)$/\3-\1-\2/' distros.txt**

SUSE 10.2 2006-12-07

Fedora 10 2008-11-25

SUSE 11.0 2008-06-19

Ubuntu 8.04 2008-04-24

([0-9]{2}/[0-9]{2}/[0-9]{4})$

которому соответствуют две цифры, слеш, две цифры, слеш, четыре цифры и ко­нец строки. Чтобы создать подвыражение, достаточно просто заключить часть регулярного выражения в круглые скобки. Первому соответствует месяц, второму — число месяца и третьему — год. Соответственно строку замены можно выразить так:

\3-\1-\2

что даст нам в результате такую последовательность: год, дефис, месяц, дефис, число месяца.

Однако существует возможность создавать более сложные команды в файлах сценариев и передавать эти сценарии с помощью параметра -f. Отчет будет содержать заголовок вверху, измененные даты и названия дис­трибутивов будут преобразованы в верхний регистр.

# Сценарий для sed, создающий отчет о дистрибутивах Linux

1 i\

\

Linux Distributions Report\

s/\([0-9]\{2\}\)\/\([0-9]\{2\}\)\/\([0-9]\{4\}\)$/\3-\1-\2/

y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/

Сохраните сценарий в файл с именем *distros.sed* и запустите его:

[me@linuxbox ~]$ **sed -f distros.sed distros.txt**

Linux Distributions Report

SUSE 10.2 2006-12-07

FEDORA 10 2008-11-25

SUSE 11.0 2008-06-19

UBUNTU 8.04 2008-04-24

Строка с номером 1 — это *комментарий*. в Linux, коммента­рии начинаются с символа #, за которым следует пояснительный текст. Строка 2 — это пустая строка. В строках с 3-й по 6-ю содержится текст, который должен быть вставлен по адре­су 1, в первую входную строку. Команда i, сопровождаемая последовательно­стью из обратного слеша и перевода строки, производит экранированный символ перевода, или то, что мы называем *символом продолжения строки*. Шестая строка в нашем сценарии фактически завершает вставляемый текст и заканчивается уже не символом продолжения строки, а про­стым переводом строки. Строка 7 — наша команда поиска с заменой. Строка 8 выполняет перекодировку букв нижнего регистра в буквы верхнего ре­гистра. Обратите внимание, что, в отличие от программы tr, команда y в sed не поддерживает ни диапазоны символов (например, [a-z]), ни классы символов POSIX.

А теперь похвастаемся умением пользоваться программой sed и вернем наши ошибки на место, чтобы продолжить эксперименты с нашим файлом:

[me@linuxbox ~]$ **sed -i 's/lazy/laxy/; s/jumped/jimped/' foo.txt**

Параметр -i сообщает программе sed, что требуется отредактировать файл «на месте», в том смысле, что изменения нужно произвести в самом файле, а не пере­слать их в стандартный вывод.

**aspell — интерактивная проверка орфографии**

Последний инструмент, который мы рассмотрим в этой главе, — программа aspell, интерактивное средство проверки орфографии.

aspell check *текстовый\_файл*

Если вызвать программу aspell без параметра --dont-backup, она создаст резерв­ную копию файла с исходным текстом, добавив к имени файла расширение *.bak*.

----------------------------------------------------------------------

1. nl — нумерует строки.
2. fold — выполняет перенос строк, ограничивая их указанной длиной.
3. fmt — выполняет простое форматирование текста.
4. pr — форматирует текст для печати.
5. printf — форматирует и выводит данные.
6. groff — система форматирования документов.

**nl — нумерация строк**

Программа nl — предназначена для решения простой задачи: она выполняет нуме­рацию строк.

[me@linuxbox ~]$ **nl distros.txt | head**

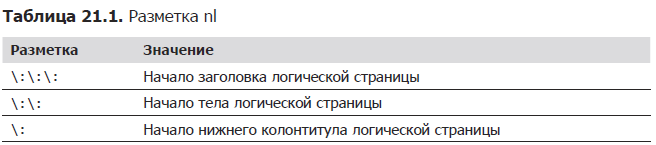
1 SUSE 10.2 12/07/2006

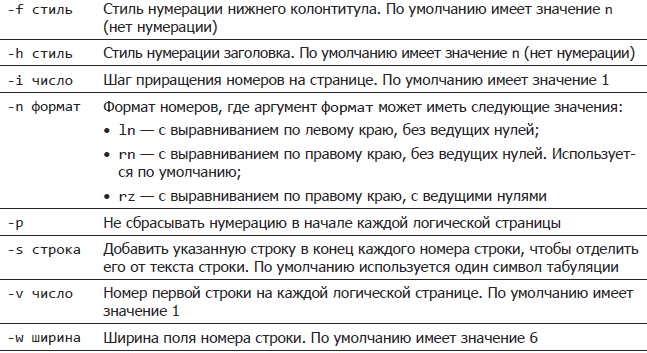
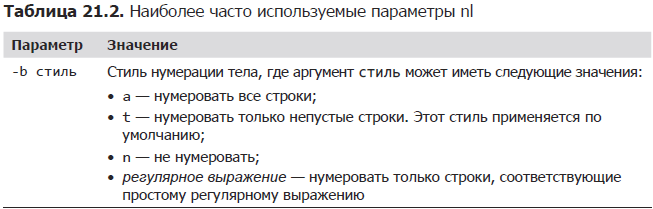
2 Fedora 10 11/25/2008

3 SUSE 11.0 06/19/2008

Так же, как cat, программа nl может принимать несколько имен файлов, однако nl имеет ряд параметров и поддерживает простейшую форму разметки, обеспечивая более сложные способы нумерации.

nl поддерживает идею *логических страниц*. Это дает возможность начинать ну­мерацию на каждой странице заново.





**fold — перенос строк после указанной длины**

*Перенос строк* заключается в разрыве текстовых строк по указанной ширине.

[me@linuxbox ~]$ **echo "The quick brown fox jumped over the lazy dog." | fold -w 12**

The quick br

own fox jump

ed over the

lazy dog.

Текст, посланный командой echo, был разбит на сегменты указанной в параметре -w ширины. Если ширина не указана, по умолчанию она принимается равной 80 символам. Обратите внимание, что стро­ки были разбиты без учета границ слов. Добавив параметр -s, можно заставить fold разбивать строки по последнему доступному пробелу перед достижением указанной ширины.

[me@linuxbox ~]$ **echo "The quick brown fox jumped over the lazy dog." | fold -w 12 -s**

The quick

brown fox

jumped over

the lazy

dog.

**fmt — простое форматирование текста**

Программа fmt также позволяет выполнить перенос строк плюс кое-что еще. Она принимает файлы или данные со стандартного ввода и формирует абзацы в по­токе текста. По сути, она заполняет и объединяет строки, сохраняя пустые строки и отступы.

Теперь представьте, что нам нужно переформатировать этот текст, уме­стив его в колонку шириной 50 символов. Решить эту задачу можно с помощью команды fmt и ее параметра –w. Добавив параметр -c, мы получили желаемый результат.

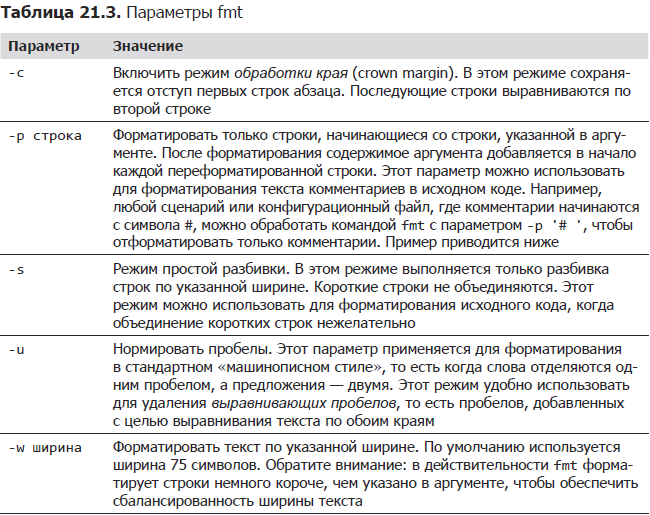
[me@linuxbox ~]$ **fmt -cw 50 fmt-info.txt**

`fmt' читает текст из файла, заданного

аргументами FILE (или со стандартного ввода,

если аргументы отсутствуют), и выводит результат

в стандартный вывод.



Особый интерес представляет параметр -p. С его помощью можно форматировать выбранные фрагменты файла, гарантировав, что все отформатированные строки будут начинаться с одной и той же последовательности символов. Файл примера содержит комментарии, начинающиеся со строки # (символ # и пробел), и строки «кода». Теперь воспользуемся командой fmt, чтобы отформа­тировать комментарии и при этом не затронуть код:

[me@linuxbox ~]$ **fmt -w 50 -p '# ' fmt-code.txt**

# Этот файл содержит код с комментариями.

# Эта строка - комментарий. За ней следует другая

# строка с комментарием. И еще одна.

Это не комментарий, а строка с кодом.

Еще одна строка с кодом.

И еще.

Обратите внимание, что смежные строки комментариев были объединены, а пу­стые строки и строки, не начинающиеся с указанного префикса, остались нетро­нутыми.

**pr — форматирование текста для печати**

Программа pr используется для *разбивки текста на страницы*. Перед печатью текст зачастую желательно разбить на страницы с несколькими пустыми строка­ми в начале и в конце, чтобы получить пустые поля вверху и внизу каждой страни­цы. В дальнейшем эти поля можно использовать для вставки верхнего и нижнего колонтитулов.



В этом примере использовались параметры -l (длина (length) страницы) и -w (ширина (width) страницы), определяющие размеры «страницы» — 65 символов в ширину и 15 строк в длину. pr разбила содержимое файла *distros.txt* на отдельные страницы, добавив несколько пустых строк сверху и снизу, и создала заголовок по умолчанию, содержащий время последнего изменения файла, имя файла и номер страницы.

**printf — форматирование и вывод данных**

В отличие от других команд в этой главе, команда printf не используется в кон­вейерной обработке (она не принимает данные со стандартного ввода) и редко применяется непосредственно в командной строке (чаще она используется в сце­нариях). Почему это так важно? Потому что она используется очень широко.

printf "*формат*" *аргументы*

[me@linuxbox ~]$ **printf "I formatted the string: %s\n" foo**

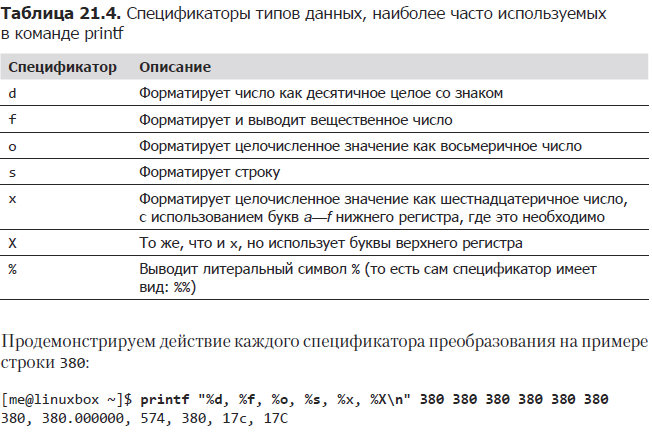
I formatted the string: foo

Строка формата может содержать *литеральный* текст (такой, как I formatted the string:), экранированные последовательности (такие, как \n, символ пере­вода строки) и последовательности, начинающиеся с символа %, которые называ­ют *спецификаторами преобразований* (conversion specifications). В примере выше спецификатор преобразования %s используется для форматирования строки foo и включения ее в вывод команды.

[me@linuxbox ~]$ **printf "I formatted '%s' as a string.\n" foo**

I formatted 'foo' as a string.

Как видите, в выводе команды спецификатор преобразования %s замещается стро­кой foo.



Дополнительная информация 🡪 Командная строка Linux. Полное руководство стр. 311

**Системы форматирования документов**

В области систем форматирования документов доминируют два основных се­мейства программ: уходящие корнями в оригинальную программу roff, включая nroff и troff

**groff — система форматирования документов**

groff — это пакет программ с GNU-реализацией troff. Он также включает сцена­рий, имитирующий работу nroff, и остальные программы семейства roff.

Дополнительная информация 🡪 Командная строка Linux. Полное руководство стр. 315

------------------------------------------------------------------------------------

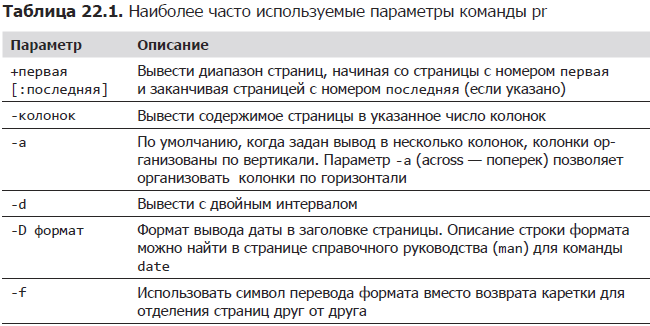
**Печать**

1. pr — преобразует текстовые файлы для печати.
2. lpr — печатает файлы.
3. lp — печатает файлы (System V).
4. a2ps — форматирует файлы для печати на принтере с поддержкой PostScript.
5. lpstat — выводит информацию о состоянии принтера.
6. lpq — выводит информацию о состоянии очереди печати.
7. lprm — отменяет задания печати.
8. cancel — отменяет задания печати (System V).

Поиск принтеров для добавления с установкой драйвера

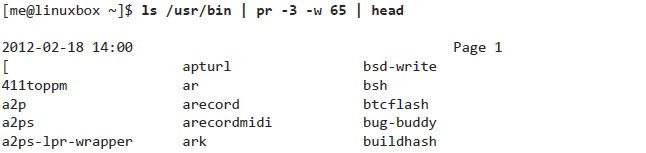
system-config-printer

**pr — преобразование текстовых файлов для печати**





Следующий при­мер создает список содержимого каталога */usr/bin* и с помощью pr выводит его в три колонки с разбивкой на страницы:



**lpr — печать файлов (в стиле Berkeley)**

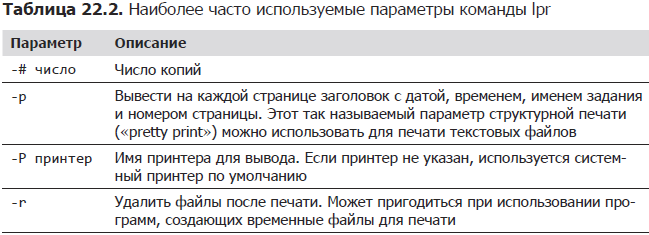
Программа lpr применяется для отправки файлов на принтер.

Для вывода файла на другой принтер используйте параметр -P:

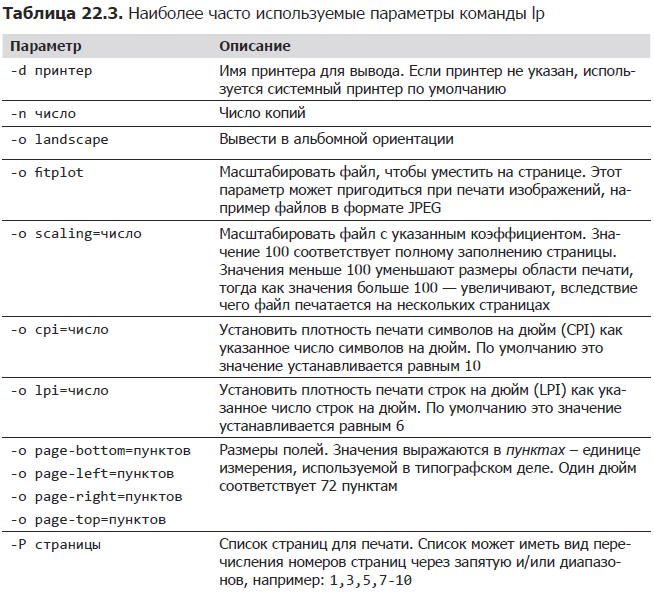
lpr -P *имя\_принтера*

где аргумент имя\_принтера — имя требуемого принтера. Получить список принте­ров, известных системе, поможет команда:

[me@linuxbox ~]$ **lpstat –a**



**lp — печать файлов (в стиле System V)**



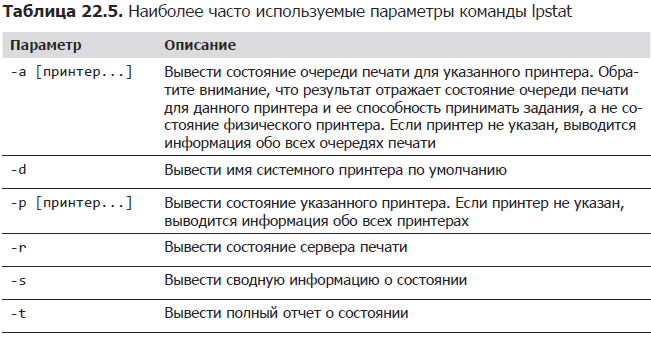
Lp *имя файла*

**lpstat — вывод информации о состоянии принтера**

[me@linuxbox ~]$ **lpstat –a**

Кроме того, с ее помощью можно получить более подробное описание конфигура­ции системы печати:

[me@linuxbox ~]$ **lpstat –s**



**lpq — вывод информации о состоянии очереди печати**

Программа lpq используется для получения информации о состоянии очереди печати.

[me@linuxbox ~]$ **lpq**

printer is ready

no entries

Если принтер не указан (с помощью параметра -P), выводится информация об очереди для системного принтера по умолчанию.

**lprm и cancel — отмена заданий печати**

[me@linuxbox ~]$ **lpq**

printer is ready and printing

Rank Owner Job File(s) Total Size

active me 603 (stdin) 1024 bytes

Если использовать пример с за­данием печати, рассматриваемый выше, мы могли бы остановить выполнение за­дания и удалить его:

[me@linuxbox ~]$ **cancel 603**

[me@linuxbox ~]$ **lpq**

printer is ready

no entries

**Компиляция программ**

1. make — утилита сопровождения программ.

После загрузки tar-файла его нужно распаковать. Делается это с помощью про­граммы tar:

[me@linuxbox src]$ **tar xzf diction-1.11.tar.gz**

При распаковыва­нии некоторых проектов каталог не создается, а файлы помещаются непосредственно в текущий каталог, что может вызвать неразбериху и путаницу в хорошо организованном каталоге *src*. Чтобы избежать этого, пользуйтесь следующей командой для исследования содержимого tar-файла:

tar tzvf tarfile | head

Дополнительная информация 🡪 Командная строка Linux. Полное руководство стр. 338

-------------------------------------------------------------------

**Как написать сценарий командной оболочки**

Итак, запустите текстовый редактор и введите следующий сценарий:

#!/bin/bash

# Это наш первый сценарий.

echo 'Hello World!'

Сохраните файл сценария с именем *hello\_world*.

[me@linuxbox ~]$ **chmod 755 hello\_world**

[me@linuxbox ~]$ **ls -l hello\_world**

-rwxr-xr-x 1 me me 63 2012-03-07 10:10 hello\_world

Она содержит список каталогов, перечисленных через двоеточие. Увидеть, что содержится в PATH, можно с помощью команды:

[me@linuxbox ~]$ **echo $PATH**

/home/me/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

В большинстве дистрибутивов Linux в переменную PATH включается каталог *bin* в домашнем каталоге пользователя, чтобы дать поль­зователям возможность выполнять собственные программы. То есть если создать каталог *bin* и поместить сценарий в него, его можно будет запускать как любые другие программы:

[me@linuxbox ~]$ **mkdir bin**

[me@linuxbox ~]$ **mv hello\_world bin**

Если каталог отсутствует в переменной PATH, его легко туда добавить, включив следующую строку в файл *.bashrc*:

export PATH=~/bin:"$PATH"

Чтобы применить изменения в текущем сеансе, нужно заставить команд­ную оболочку повторно прочитать файл *.bashrc*, например, так:

[me@linuxbox ~]$ **. .bashrc**

Команда «точка» (.) является синонимом source.

Сценарии, которые должны быть доступны всем пользователям в системе, лучше размещать в традиционном местоположении — в каталоге */usr/local/bin*. Сценарии, предназначенные для использования си­стемным администратором, часто помещаются в каталог */usr/local/sbin*. В боль­шинстве случаев программное обеспечение, созданное в локальной системе, будь то сценарии или скомпилированные программы, следует помещать в иерархию каталогов */usr/local*, а не */bin* или */usr/bin*.

Дополнительная информация 🡪 Командная строка Linux. Полное руководство стр. 353

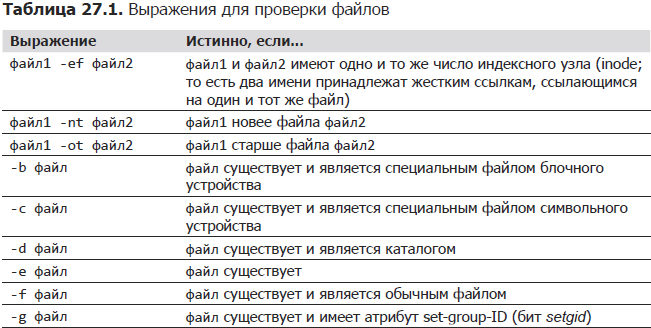
---------------------------------------------------------------------

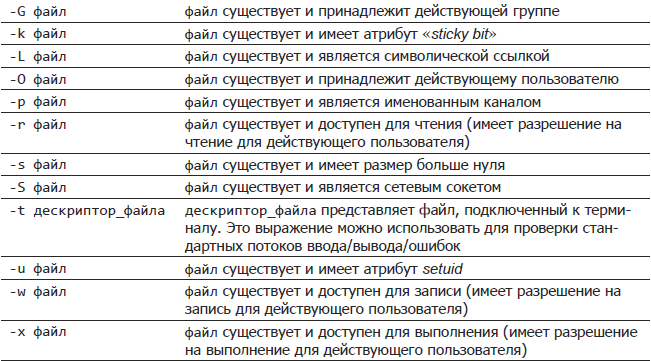
**Команда test**

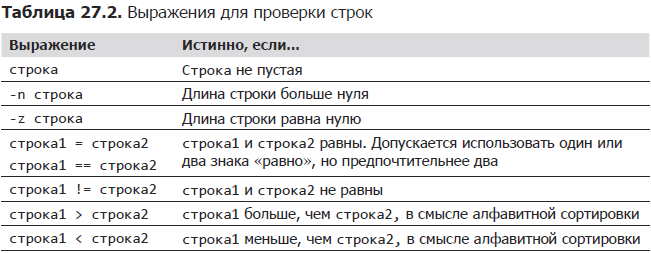
Команда test может выполнять различные проверки и сравнения.

test *выражение*

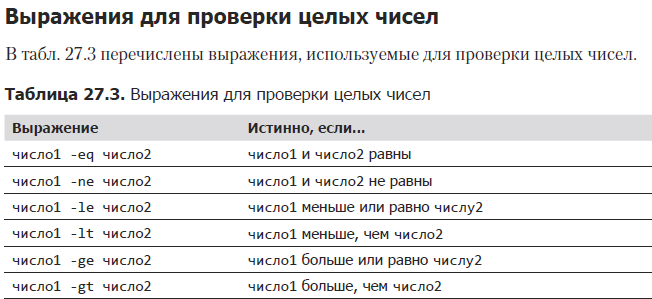
Команда test возвращает код завершения 0, если выражение истинно, и код завершения 1, если выражение ложно.







При использовании с командой test операторы > и < необходимо заключать в кавычки (или экранировать символом обратного слеша). Если этого не сделать, они будут ин­терпретироваться командной оболочкой как операторы перенаправления, что может привести к плачевным результатам.



Следующий сценарий демонстрирует их применение:

#!/bin/bash

# test-integer: проверка целочисленного значения.

INT=-5

if [ -z "$INT" ]; then

echo "INT is empty." >&2

exit 1

fi

if [ $INT -eq 0 ]; then

echo "INT is zero."

else

if [ $INT -lt 0 ]; then

echo "INT is negative."

else

echo "INT is positive."

fi

if [ $((INT % 2)) -eq 0 ]; then

echo "INT is even."

else

echo "INT is odd."

fi

fi

**Более современная версия команды test**

Она имеет следующий синтаксис:

[[ *выражение* ]]

где выражение возвращает истинное (true) или ложное (false) значение.

#!/bin/bash

# test-integer2: проверка целочисленного значения.

INT=-5

**if [[ "$INT" =~ ^-?[0-9]+$ ]]; then**

if [ $INT -eq 0 ]; then

echo "INT is zero."

else

if [ $INT -lt 0 ]; then

echo "INT is negative."

else

echo "INT is positive."

fi

if [ $((INT % 2)) -eq 0 ]; then

echo "INT is even."

else

echo "INT is odd."

fi

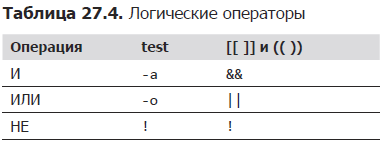
fi

**else**

**echo "INT is not an integer." >&2**

**exit 1**

**fi**

****

**(( )) — для проверки целых чисел**

Команда (( )) применяется для *проверки истинности арифметических выраже­ний*. Арифметическое выражение считается истинным, если его результат отлича­ется от нуля.

[me@linuxbox ~]$ **if ((1)); then echo "It is true."; fi**

It is true.

[me@linuxbox ~]$ **if ((0)); then echo "It is true."; fi**

[me@linuxbox ~]$

#!/bin/bash

# test-integer2a: проверка целочисленного значения.

INT=-5

if [[ "$INT" =~ ^-?[0-9]+$ ]]; then

if **((INT == 0));** then

echo "INT is zero."

else

if **((INT < 0));** then

echo "INT is negative."

Else

echo "INT is positive."

fi

if **(( ((INT % 2)) == 0));** then

echo "INT is even."

else

echo "INT is odd."

fi

fi

else

echo "INT is not an integer." >&2

exit 1

fi

**Операторы управления: еще один способ ветвления**

bash поддерживает два оператора управления

*команда1* && *команда2*

и

*команда1* || *команда2*

[me@linuxbox ~]$ **mkdir temp && cd temp**

Она создаст каталог с именем *temp* и, если эта операция завершится успехом, ката­лог *temp* будет назначен текущим рабочим каталогом.

[me@linuxbox ~]$ **[ -d temp ] || mkdir temp**

проверит существование каталога *temp*, и только если проверка не увенчается успехом, будет выполнена команда его создания.

[ -d temp ] || exit 1

Если сценарий требует наличия каталога *temp*, а он не существует, тогда сценарий завершится с кодом 1.

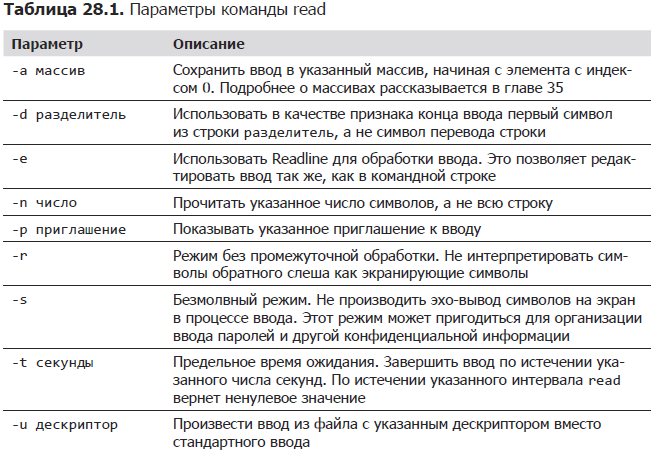
----------------------------------------------------------------------------

**read — чтение значений со стандартного ввода**

Встроенная команда read используется для чтения единственной строки со стан­дартного ввода.

read [-*параметры*] [*переменная*...]

где параметры — это один или несколько параметров из перечисленных в табл. 28.1, а переменная — имя одной или нескольких переменных для сохранения введен­ного значения.



#!/bin/bash

# read-integer: проверка целочисленного значения.

**echo -n "Please enter an integer -> "**

**read int**

if [[ "$int" =~ ^-?[0-9]+$ ]]; then

if [ $int -eq 0 ]; then

echo "$int is zero."

else

if [ $int -lt 0 ]; then

echo "$int is negative."

else

echo "$int is positive."

fi

if [ $((int % 2)) -eq 0 ]; then

echo "$int is even."

else

echo "$int is odd."

fi

fi

else

echo "Input value is not an integer." >&2

exit 1

fi

Сначала мы использовали команду echo с параметром -n (подавляющим вы­вод символа перевода строки в конце) для вывода приглашения к вводу, а затем команду read для ввода значения в переменную int.

[me@linuxbox ~]$ **read-integer**

Please enter an integer -> **5**

5 is positive.

5 is odd.

Дополнительная информация 🡪 Командная строка Linux. Полное руководство стр. 389

Проверка кода завершения [me@linuxbox ~]$ echo $?

-----------------------------------------------------------------------------------------------

**While**

#!/bin/bash

# while-menu2: программа вывода системной информации,

# управляемая с помощью меню

DELAY=3 # Время отображения результатов на экране (в секундах)

**while true; do**

clear

cat <<- \_EOF\_

Please Select:

1. Display System Information

2. Display Disk Space

3. Display Home Space Utilization

0. Quit

\_EOF\_

read -p "Enter selection [0-3] > "

if [[ $REPLY =~ ^[0-3]$ ]]; then

if [[ $REPLY == 1 ]]; then

echo "Hostname: $HOSTNAME"

uptime

sleep $DELAY

**continue**

fi

if [[ $REPLY == 2 ]]; then

df -h

sleep $DELAY

**continue**

fi

if [[ $REPLY == 3 ]]; then

if [[ $(id -u) -eq 0 ]]; then

echo "Home Space Utilization (All Users)"

du -sh /home/\*

else

echo "Home Space Utilization ($USER)"

du -sh $HOME

fi

sleep $DELAY

**continue**

fi

**if [[ $REPLY == 0 ]]; then**

**break**

**fi**

else

echo "Invalid entry."

sleep $DELAY

fi

done

echo "Program terminated."

**Трассировка**

Чтобы увидеть, как в действительности протекает выполнение программы, воспользуемся при­емом *трассировки*.

Один из способов трассировки заключается в размещении информативных со­общений в разных точках сценария, сообщающих, где протекает выполнение.

**echo "preparing to delete files" >&2**

if [[ -d $dir\_name ]]; then

if cd $dir\_name; then

**echo "deleting files" >&2**

rm \*

else

echo "cannot cd to '$dir\_name'" >&2

exit 1

fi

else

echo "no such directory: '$dir\_name'" >&2

exit 1

fi

**echo "file deletion complete" >&2**

Теперь, запустив сценарий, убедимся, что удаление файлов действительно было выполнено:

[me@linuxbox ~]$ **deletion-script**

preparing to delete files

deleting files

file deletion complete

[me@linuxbox ~]$

Кроме того, bash поддерживает встроенный метод трассировки, реализованный в виде параметра -x и команды set с параметром -x.

**#!/bin/bash -x**

# trouble: сценарий для демонстрации распространенных видов ошибок

number=1

if [ $number = 1 ]; then

echo "Number is equal to 1."

else

echo "Number is not equal to 1."

fi

После запуска мы получим следующие результаты:

[me@linuxbox ~]$ **trouble**

+ number=1

+ '[' 1 = 1 ']'

+ echo 'Number is equal to 1.'

Number is equal to 1.

--------------------------------------------------------------------

**case**

Командная оболочка bash поддерживает составную команду выбора из несколь­ких вариантов, которая называется case.

С помощью case можно сделать логику выбора немного проще:

#!/bin/bash

# case-menu: программа вывода системной информации,

# управляемая с помощью меню

clear

echo "

Please Select:

1. Display System Information

2. Display Disk Space

3. Display Home Space Utilization

0. Quit

"

read -p "Enter selection [0-3] > "

**case $REPLY in**

**0) echo "Program terminated."**

**exit**

**;;**

**1) echo "Hostname: $HOSTNAME"**

**uptime**

**;;**

**2) df -h**

**;;**

**3) if [[ $(id -u) -eq 0 ]]; then**

**echo "Home Space Utilization (All Users)"**

**du -sh /home/\***

**else**

**echo "Home Space Utilization ($USER)"**

**du -sh $HOME**

**fi**

**;;**

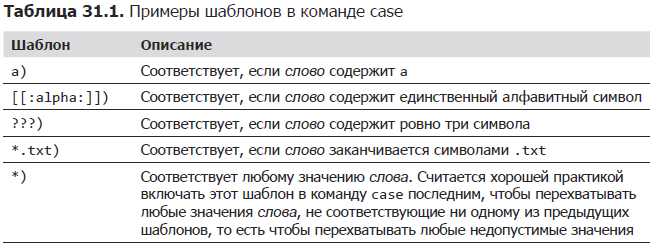
**\*) echo "Invalid entry" >&2**

**exit 1**

**;;**

**esac**

Найдя соответствие, она выполняет *команды*, связанные с найденным шаблоном. После нахождения соответствия сопоставление с нижележащими шаблонами уже не производится.



--------------------------------------------------------------------------------------

**for: традиционная форма**

for *переменная* [in *слова*]; do

*команды*

done

где переменная — это имя переменной, значение которой будет увеличиваться в ходе выполнения цикла, слова — необязательный список элементов, которые последовательно будут присваиваться переменной, и команды — это команды, вы­полняемые в каждой итерации.

[me@linuxbox ~]$ **for i in A B C D; do echo $i; done**

A

B

C

D

Например, можно использовать подстановку в фигурных скобках:

[me@linuxbox ~]$ **for i in {A..D}; do echo $i; done**

A

B

C

D

или подстановку имен файлов:

[me@linuxbox ~]$ **for i in distros\*.txt; do echo $i; done**

distros-by-date.txt

distros-dates.txt

distros-key-names.txt

**for: форма в стиле языка C**

for (( *выражение1*; *выражение2*; *выражение3* )); do

*команды*

done

где выражение1, выражение2 и выражение3 — это арифметические выражения, а команды — это команды, выполняемые в каждой итерации цикла.

-----------------------------------------------------------------------------

**Получение имен переменных**

${!*префикс*\*}

${!*префикс*@}

Эти две формы подстановки возвращают имена существующих переменных, на­чинающиеся с указанного префикса.

Следующая команда выводит список всех пе­ременных окружения с именами, начинающимися с BASH:

[me@linuxbox ~]$ **echo ${!BASH\*}**

BASH BASH\_ARGC BASH\_ARGV BASH\_COMMAND BASH\_COMPLETION BASH\_COMPLETION\_DIR BASH\_LINENO BASH\_SOURCE BASH\_SUBSHELL BASH\_VERSINFO BASH\_VERSION

**Битовые операции**

Они применяются для выполнения некоторых низкоуровневых операций, часто связанных с установкой или чтением битовых флагов.

Ниже показано, как с использованием оператора поразрядного сдвига влево вы­вести список степеней 2

[me@linuxbox ~]$ **for ((i=0;i<8;++i)); do echo $((1<<i)); done**

1

2

4

8

16

32

64

128

-----------------------------------------------------------------

**Создание массива**

Например:

[me@linuxbox ~]$ **a[1]=foo**

[me@linuxbox ~]$ **echo ${a[1]}**

foo

Это пример присваивания значения элементу массива и обращения к нему. Пер­вая команда присваивает значение foo элементу массива a с индексом 1. Вторая команда выводит значение.

Массив можно также создать командой declare:

[me@linuxbox ~]$ **declare -a a**

Параметр -a в этом примере требует от declare создать массив (array) с именем a.

**Присваивание значений массиву**

*имя*[*индекс*]=*значение*

где имя — это имя массива, индекс — целое число (или арифметическое выраже­ние) больше или равное 0.

Напри­мер, если понадобится присвоить элементам массива days сокращенные названия дней недели, это можно сделать так:

[me@linuxbox ~]$ **days=(Sun Mon Tue Wed Thu Fri Sat)**

Можно присваивать значения конкретным элементам, указывая индекс для каж­дого значения:

[me@linuxbox ~]$ **days=([0]=Sun [1]=Mon [2]=Tue [3]=Wed [4]=Thu [5]=Fri [6]=Sat)**

**Доступ к элементам массива**

Рассмотрим простой пример сбора и представления данных.

В этом примере мы запустили программу hours, передав ей текущий каталог для анализа. Она вывела таблицу, показывающую число файлов, изменявшихся Доступ к элементам массива **463**

в каждый час суток (0–23).

#!/bin/bash

# hours : сценарий для подсчета файлов по времени изменения

usage () {

echo "usage: $(basename $0) directory" >&2

}

# Убедиться, что аргумент является каталогом

if [[ ! -d $1 ]]; then

usage

exit 1

fi

# Инициализировать массив

for i in {0..23}; do hours[i]=0; done

# Собрать данные

for i in $(stat -c %y "$1"/\* | cut -c 12-13); do

j=${i/#0}

((++hours[j]))

((++count))

done

# Вывести данные

echo -e "Hour\tFiles\tHour\tFiles"

echo -e "----\t-----\t----\t-----"

for i in {0..11}; do

j=$((i + 12))

printf "%02d\t%d\t%02d\t%d\n" $i ${hours[i]} $j ${hours[j]}

done

printf "\nTotal files = %d\n" $count

Дополнительная информация 🡪 Командная строка Linux. Полное руководство стр. 463

**Поиск используемых индексов**

Так как bash позволяет создавать разреженные массивы путем присваивания зна­чений отдельным элементам, иногда требуется определить, какие элементы дей­ствительно существуют.

${!*массив*[\*]}

${!*массив*[@]}

где массив — это имя переменной-массива. Как и в других случаях использова­ния \* и @ в операциях подстановки, форма @, заключенная в кавычки, оказывается наиболее полезной, так как выполняет подстановку нераздробленных значений элементов:

[me@linuxbox ~]$ **foo=([2]=a [4]=b [6]=c)**

[me@linuxbox ~]$ **for i in "${foo[@]}"; do echo $i; done**

a

b

c

[me@linuxbox ~]$ **for i in "${!foo[@]}"; do echo $i; done**

2

4

6

**Добавление элементов в конец массива**

Знание количества элементов в массиве не поможет, если понадобится добавить значения в конец массива, потому что значения, возвращаемые индексами \* и @, не сообщают наибольший занятый индекс в массиве. Оператор присваивания += автоматически добавляет значения в конец массива.

[me@linuxbox ~]$ **foo=(a b c)**

[me@linuxbox ~]$ **echo ${foo[@]}**

a b c

[me@linuxbox ~]$ **foo+=(d e f)**

[me@linuxbox ~]$ **echo ${foo[@]}**

a b c d e f

**Сортировка массива**

#!/bin/bash

# array-sort : сортировка массива

a=(f e d c b a)

echo "Original array: ${a[@]}"

a\_sorted=($(for i in "${a[@]}"; do echo $i; done | sort))

echo "Sorted array: ${a\_sorted[@]}"

Если запустить этот сценарий, он выведет следующее:

[me@linuxbox ~]$ **array-sort**

Original array: f e d c b a

Sorted array: a b c d e f

**Удаление массива**

Удалить массив можно с помощью команды unset:

[me@linuxbox ~]$ **foo=(a b c d e f)**

[me@linuxbox ~]$ **echo ${foo[@]}**

a b c d e f

[me@linuxbox ~]$ **unset foo**

Командой unset можно также удалить единственный элемент массива:

[me@linuxbox ~]$ **foo=(a b c d e f)**

[me@linuxbox ~]$ **echo ${foo[@]}**

a b c d e f

[me@linuxbox ~]$ **unset 'foo[2]'**

Интересно отметить, что присваивание пустого значения массиву не уничтожает его содержимое:

[me@linuxbox ~]$ **foo=(a b c d e f)**

[me@linuxbox ~]$ **foo=**

[me@linuxbox ~]$ **echo ${foo[@]}**

b c d e f

Любая ссылка на переменную-массив без индекса возвращает элемент с индек­сом 0:

[me@linuxbox ~]$ **foo=(a b c d e f)**

[me@linuxbox ~]$ **echo ${foo[@]}**

a b c d e f

[me@linuxbox ~]$ **foo=A**

[me@linuxbox ~]$ **echo ${foo[@]}**

A b c d e f

------------------------------------------------------------------------------------------

**Ловушки**

Проектируя большие и сложные сценарии, важно предусматривать их реакцию на неожиданный выход пользователя из системы или выключение компьютера во время их выполнения. Если возникают подобные события, всем процессам посылается сигнал.

trap *аргумент сигнал* [*сигнал*...]

где аргумент — это строка, которая будет прочитана и выполнена как команда, а сигнал — идентификатор сигнала, в ответ на который будет выполнена указан­ная команда.

Рассмотрим простой пример:

#!/bin/bash

# trap-demo : простой пример обработки сигналов

trap "echo 'I am ignoring you.'" SIGINT SIGTERM

for i in {1..5}; do

echo "Iteration $i of 5"

sleep 5

done

Ниже показано, как выглядят попытки остановить сценарий нажатием комбина­ции CTRL+C:

[me@linuxbox ~]$ **trap-demo**

Iteration 1 of 5

Iteration 2 of 5

I am ignoring you.

Iteration 3 of 5

I am ignoring you.

Iteration 4 of 5

Iteration 5 of 5

Как видите, каждый раз, когда пользователь пытается прервать работу програм­мы, она вместо этого выводит сообщение.

**Wait**

#!/bin/bash

# async-parent : пример асинхронного выполнения (родитель)

echo "Parent: starting..."

echo "Parent: launching child script..."

async-child &

pid=$!

echo "Parent: child (PID= $pid) launched."

echo "Parent: continuing..."

sleep 2

echo "Parent: pausing to wait for child to finish..."

wait $pid

echo "Parent: child is finished. Continuing..."

echo "Parent: parent is done. Exiting."

и дочерний сценарий:

#!/bin/bash

# async-child : пример асинхронного выполнения (потомок)

echo "Child: child is running..."

sleep 5

echo "Child: child is done. Exiting."

В этом примере дочерний сценарий тривиально прост. Фактическая работа вы­полняется родителем. Родительский сценарий запускает дочерний сценарий и пе­реводит его в фоновый режим выполнения. Идентификатор дочернего процесса сохраняется в переменной pid путем присваивания ей значения параметра $!, ко­торый всегда содержит идентификатор процесса последнего задания, переведен­ного в фоновый режим.

Родительский сценарий продолжает работу и в конце выполняет команду wait с идентификатором процесса дочернего сценария. Это вызывает приостановку родительского сценария до завершения дочернего сценария, после чего родитель­ский сценарий возобновляет работу и тут же завершается.

---------------------------------------------------------------------------------------------

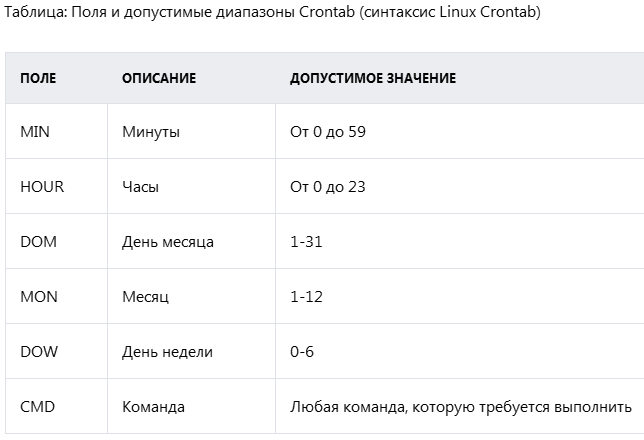
## Как работает Cron?

Аналог планировщик задач на windows.

Фактически, Cron - это сервис, как и большинство других сервисов Linux, он запускается при старте системы и работает в фоновом режиме.

уществует несколько конфигурационных файлов, из которых он берет информацию о том что и когда нужно выполнять. Сервис открывает файл /etc/crontab, в котором указаны все нужные данные. Часто, в современных дистрибутивах там прописан запуск утилиты run-parts, которая запускает нужные скрипты из следующих папок:

* **/etc/cron.minutely** - каждую минуту;
* **/etc/cron.hourly** - каждый час;
* **/etc/cron.daily** - каждый день;
* **/etc/cron.weekly** - каждую неделю;
* **/etc/cron.monthly** - каждый месяц.



Посмотреть информацию о всех командах запускаемых демоном cron можно в каталоге /var/log называются cron, cron1 и т.д.

Создать задание можно с помощью команды :

$ crontab

После сохранения она попадает в /tmp/тут\_название.

Дата выполнения задания последний раз записывается в файл /var/spool/anacron, а затем, при следующем запуске anacron проверяет был ли запущен нужный процесс в нужное время, и если нет, то запускает его.

команда crontab:

$ crontab –e

Сначала можно посмотреть задачи cron, для этого можно воспользоваться опцией -l:

$ crontab –l

Вы можете удалить все существующие задачи командой -r:

$ crontab –r

-u означает чей crontab будет обработан, Если опция не задана, то будет обработан crontab того пользователя, который запустил команду crontab.

-------------------------

crontab -u user\_name -l -просмотр файла crontab

------------------------- юзера "user\_name"

-------------------------

crontab -u user\_name -r -удаление файла crontab

------------------------- юзера "user\_name"

-------------------------

crontab -u user\_name -e -редактирование файла crontab

------------------------- юзера "user\_name" используя

редактор, заданный переменной

окружения VISUAL или EDITOR

Самый простой пример - запускать его каждую минуту:

\* \* \* \* \* /usr/local/bin/serve

Или

0-59 \* \* \* \* /home/user/mail

Запускаем в нулевую минуту нулевого часа, каждый день, это в 12 ночи:

0 0 \* \* \* /usr/local/bin/serve

Можно в любой день, например, 15 числа:

0 0 15 \* \* /usr/local/bin/serve

Вы можете выбрать любую минуту, час и день недели, например, 15.30 во вторник:

30 15 \* \* 2 /usr/local/bin/serve

Для того чтобы указать определенный интервал нужно использовать символ "-", например, каждый час, с семи утра до семи вечера:

0 7-19  \* \* \* /usr/local/bin/serve

Если нужно запустить команду несколько раз, можно использовать разделитель ",". Например, запустим скрипт в 5 и 35 минут пятого (16:05 и 16:35), каждый день:

5,35 16  \* \* \* /usr/local/bin/serve

Вы можете захотеть не указывать отдельно время, а просто указать интервал, с которым нужно запускать скрипт, например, раз в 10 минут. Для этого используется разделитель косая черта - "/":

\*/10 \* \* \* \* /usr/local/bin/serve

Кроме того, для некоторых часто используемых наборов были придуманы переменные, вот они:

* **@reboot** - при загрузке, только один раз;
* **@yearly, @annually** - раз год;
* **@monthly** - раз в месяц;
* **@weekly** - раз в неделю;
* **@daily, @midnight** - каждый день;
* **@hourly** - каждый час.

После того как вы настроили правила, еще хотелось бы проверить работают ли они. Для этого ждем того времени, когда скрипт уже должен быть выполнен и смотрим лог cron. Иногда он находится в /var/log/cron, а иногда пишется в syslog.

\* \*/2 \* \* sat echo "HELLO" -печать HELLO каждый четный час, каждую субботу.

Запуск программы с графической оболочкой (GUI); каждую минуту будет отображаться сообщение с текстом «It work!»:

0-59 \* \* \* \* DISPLAY=:0 gdialog --msgbox "It work\!" 25 20 > /dev/null

\* где DISPLAY=:0 - номер монитора

# Запуск Thunderbird каждый день в 18:30

30 18 \* \* \* export DISPLAY=:0 && thunderbird

# Закрытие Thunderbird в 19:00

00 19 \* \* \* killall thunderbird

How do I fix this problem under Fedora / CentO / RHEL or any Linux distro?

Run an ls -l on /etc/cron.d/vnstat file and find out the permissions. cron do not like files with world writable file and it is pretty picky about it. All you have to do it set 0644 (rw-r–r) permission i.e. remove world writable file permissions. Type the following command as root user:

# ls -l /etc/cron.d/vnstat  
# chmod 0644 /etc/cron.d/vnstat  
# ls -l /etc/cron.d/vnstat

This should fix the problem.

--------------------------------------------------------------------------------------------

pv - Для того, чтобы наблюдать за ходом передачи файла (не каталог), воспользуйтесь командой pv:

## Просмотр сведений о ходе передачи файла:

$ pv access.log | gzip > access.log.gz

611MB 0:00:11 [58.3MB/s] [=> ] 15% ETA 0:00:59

----------------------------------------------------------------------------------------------------------

grep – команду можно использовать для поиска слова

Найти где есть слово sergey

sergey@LinuxTest:~$ grep sergey /etc/\* 2>\error.txt

вывод только того где есть слово sergey, а ошибки сохраняются в файл error.txt

sergey@LinuxTest:~$ grep sergey /etc/\* 2> /dev/null

вывод только того где есть слово sergey, а ошибки просто игнорируются.

sergey@LinuxTest:~$ grep sergey /etc/\* > good.txt 2> nogood.txt

записать в файл good.txt только то, где есть слово sergey без ошибок, а ошибки в nogood.txt

sergey@LinuxTest:~$ grep sergey /etc/\* &> allgood.txt

сохранить в файл всё и хорошее и ошибки.