1.1 Основные команды

# ls -lash /home # Просмотр содержимого католога

# pwd # Путь к текущему каталогу

# su - # стать root + его переменные окружения

# cd /home # Переход в другой каталог

# touch index.htm # Создать новый файл

# mkdir -p /home/name/www/{tools,i} # Создать каталог с подкаталогами

# cp www/index.htm . # Копируем файл в текущий каталог

# cp -la /dir1 /dir2 # Архивирование каталога с подкаталогами

# cp -R /home/name /home/name\_bak # Копируем каталог

# cp index.htm{,.bak} # Копируем файл с новым расширением, быстрый backup

# mv -v /home/name\_bak /home/name # Перемещение/переименование файлов и каталогов

# rm -Rf /home/name/www # Удаление каталога со всем содержимым

# shred /home/name/www/\* # Удаление с перезаписью случайными числами 25 раз

# which ls # Полный путь имени/расположение команды

# whatis ls # Очень короткая справка о команде

# whereis grep # Путь к исполняемым файлам, исходным файлам и справочному рук-ву

# ldd /bin/grep # Список необходимых библиотек для работы команды

# ldconfig -n /path/to/libs/ # Добовляем путь к библиотеке(настройка динамического связывания)

# date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

# date 031713402011 # Устанавливаем дату: Чтв Мар 17 13:40:00 MSK 2011

# time ls -lash /root # Время выполнение команды

# cal -3 # Календарь на 3 месяца

# set | grep $USER # Список переменных окружения

# openssl md5 index.htm.bak # Делаем md5 hash файла

1.2 Объединение команд

# cd /home/name; ls -la # ; - последовательное выполнение команд

# ls file.txt && echo "DATA" >>file.txt # && - выполнение команды при условии успешного завершения предыдущих

# ls file1.txt || echo "DATA" >file1.txt # || - выполнение команды при условии, что предыдущая завершилась с ошибкой

# touch $(echo $(date "+%Y-%m-%d").txt) # $() - использование выходных данных одной команды при вызове другой команды

# ls -la /usr/bin | more # | - передача выходных данных одной команды на вход другой команды

# ls > menu.txt # > - перенаправление выходных данных в файл

# wc -l < menu.txt # < - использование содержимого файла в качестве входных данных

# find / -name .profile 2>&1 |less # 2>&1 - поток ошибок туда, куда направлен вывод

stdin | 0 | поток ввода | клавиатура  
stdout | 1 | поток вывода | терминал  
stdin | 2 | поток ошибок | терминал  
  
1.3 Специальные символы (групповые операции)

# rm file\*.\*; rm -Rf /home/name/www/\* # \* - любое количество(в том числе нулевое) любых символов

# rm foto1?.jpg # ? - один произвольный символ

# rm foto[12].[jpgpnif]; rm foto[3-6].jpg# [] - определенный набор символов

# mkdir -p /home/name/www/{tools,i} # {} - определить множество

В именах файлов нельзя использовать:

/ - использовать запрещено

- - нельзя ипользовать в начале имени файла или каталога

{}, \*, ?, ', ", [ , ], \, >, <, |, &, пробел - каждый из этих символовдолжен быть предварен \. Применять не рекомендуется.

# rm -Rf Рабочий\ стол # удаляем папку - Рабочий стол

# rm "Рабочий стол"; rm \[13\]foto.jpg # удаляем [13]foto.jpg

В конфигурационных файлах:

# - комментарий

// - комментарий

1.4 Просмотр содержимого файлов

# cat .bashrc; cat index.htm i/index.htm # вывод содержимого файла\файлов

# less -N /etc/named.conf # постраничный вывод текста с нумерацией строк (с прокруткой файла)

# head -t 20 /etc/named.conf # вывод первых 20 строк файла

# tail -f -n 100 /var/log/messages # вывод последних 100 строк + вывод добавленных строк в реальном времени

# more /etc/named.conf # вывод содержимого файла на экран отдельными страницами

# nano /etc/named.conf # просмотр и редактирование файла

1.5 nano  
# nano /etc/hosts # просмотр и редактирование файла

# export EDITOR=nano # делаем nano редактором файлов по умолчанию

Ctrl-X - закрыть редактор

Ctrl-O - сохранить

Ctrl-C - номер строки\текущая позиция

Ctrl-W - поиск

Ctrl-W затем Ctrl-T - переход к строке №

Ctrl-K - вырезать строку

Ctrl-U - вставить из буфера

Alt-A - выделение (вне X)

Alt-6 - копировать в буфер

2 Система

2.1 Загрузка

Последовательность при загрузке:

Инициализация BIOS => Загрузчик (grub) => Инициализация ядра (kernel initialization) => выполнение init -> {/etc/rc.d/rc.sysinit, /etc/rc.d/rc, /etc/rc.d/rc[0-6].d/, /etc/rc.d/rc.local} => virtual consoles => X

=> Инициализация BIOS:

Определение переферийных устройств и устройств для загрузки. BIOS читает и выполняет инструкцию расположенную в первом секторе загрузочного устройства. Обычно это первые 512 bytes жесткого диска.

=> Загрузчик:

Первоначальный загрузчик находит и загружает программу загрузки 2 этапа (stage 2) и передаёт ей управление (grub); используется BIOS API; обычное место загрузчика 2 этапа - /boot/. Загрузчик 2 этапа выбирает, находит и загружает ядро и RAM диск (initrd) в память; grub читает конфигурацию из /boot/grub/grub.conf. Подробнее о grub.

=> Инициализация ядра:

Обнаружение устройств. Инициализация драйверов устройств. Монтирование корневой файловой системы в режиме только чтения (read-only). Запуск процесса init.

=> Выполнение init:

init читает /etc/inittab (как загружать систему для каждого runlevel). Определяется run level, директории для запуска скриптов, стартует /etc/rc.d/rc.sysinit: udev, selinux, устанавливаются параметры ядра в /etc/sysctl.conf, устанавливаются часы, загружается keymap, подключается swap, устанавливается hostname, проверяется и перемонтируется корневая файловая система в режиме чтения-записи.  
  
2.2 Hardware

Версия системы, дистрибутив

# uname -a # Версия ядра (kernel version)

# lsb\_release -a # Информация о дистрибутиве

# cat /etc/redhat-release # Информация о версии Centos/Redhat

# cat /etc/debian\_version # Информации о версии Debian

# uptime # Как долго система работает

# hostname # Имя системы (hostname)

# man hier # Информация о системных каталогах, справка

# last reboot # История перезагрузок системы  
  
Оборудование определенное ядром

# dmesg # Обнаруженные устройства и сообщения выводимые при загрузке

# lsdev # информация об установленном оборудовании

# dd if=/dev/mem bs=1k skip=768 count=256 2>/dev/null | strings -n 8 # Читаем BIOS

# cat /proc/cpuinfo # информация о CPU

# cat /proc/meminfo # информация о RAM и swap

# watch -n1 'cat /proc/interrupts' # Просмотр изменений прерываний

# cat /proc/devices # Конфигурация устройств

# lspci -tv # Смотреть PCI устройства

# lsusb -tv # Смотреть USB устройства

# lshal # Смотрим список устройств и их свойства

# dmidecode # Смотреть DMI/SMBIOS: hardware информация из BIOS

# grep HIGHMEM /boot/config-$(uname -r) # Узнать максимально возможный размер RAM в системе  
# grep --color=tty pae /proc/cpuinfo # если pae-то с ним будет работать xen; vmx(intel),svm(AMD)-для полной виртуализации

2.3 Ресурсы и статистика

Ресурсы RAM и HDD

# free -m # Используемая и свободная оперативная память (-m for MB)

# grep MemTotal /proc/meminfo # Информация о RAM

# df # Показать свободное пространство на дисках и смонтированные устройства

# du -sh \* # Размер каталога (листинг файлов и подкаталогов)

# du -csh # Размер текущей директории

# du -ks \* | sort -n -r # Сортировать по размеру (kbytes)

# ls -lSr # Показать файлы с сортировкой по размеру

Статистика

# top # Статистика процессов cpu

# htop # Текущее состояние системы

# iotop # Анализа потребления дисковой полосы

# atop # Статистика загрузки cpu, блочных устройств, сетевых интерфейсов

# iftop # Наблюдение за трафиком в реальном времени (сеть)

# powertop # Показывает время, затраченное на обслуживание процессором

# dnstop # Анализатор DNS-трафика на интерфейсе

# iptstate # Монитор contrack из iptables, показывает активные трансляции с возможностью их убить

# mpstat 1 # Статистика загрузки процессора

# vmstat 2 # статистика использования виртуальной памяти

# iostat 2 # Статистика I/O (2 s интервал)

# tail -n 500 /var/log/messages # Последнии 500 kernel/syslog сообщений

# tail /var/log/warn # Предупреждения об ошибках, см. syslog.conf

2.4 Ограничения

Ограничения shell/script

# ulimit -a # Смотрим ограничения

# ulimit -n 10240 # Меняем ограничение на количество открытых файлов, только shell

Ограничения user/process

# cat /etc/security/limits.conf  
\* hard nproc 250 # Ограничения пользовательских процессов

asterisk hard nofile 409600 # Ограничения на открытые файлы приложения

Ограничения на всю систему

# sysctl -a # Смотрим системные ограничения

# sysctl fs.file-max # Максимальное количество открытых файлов

# sysctl fs.file-max=102400 # Меняем максимальное каличество открытых файлов

# echo "1024 50000" > /proc/sys/net/ipv4/ip\_local\_port\_range # диапазон портов

# cat /etc/sysctl.conf

fs.file-max=102400 # Ввод по умолчанию sysctl.conf

# cat /proc/sys/fs/file-nr # Сколько дескрипторов файлов используется

Отключаем ответы сервера на ping

# sysctl net.ipv4.icmp\_echo\_ignore\_all # проверяем значение

# sysctl -w net.ipv4.icmp\_echo\_ignore\_all=1 # отключаем ответ на ping

# чтоб сохранялось при перегрузке

# nano /etc/sysctl.conf

net.ipv4.icmp\_echo\_ignore\_all=1 # выставляем 1 и сохраняем  
  
2.5 Runlevels

Ядро стартует init, init читает /etc/inittab и запускает rc, который в свою очередь стартует скрипты определенного уровня (runlevel). Скрипты находятся в /etc/init.d и ссылаются (линкуются) с /etc/rc.d/rcN.d где N номер runlevel

Уровень запуска по умолчанию указывается в /etc/inittab, обычно это 3 или 5.

# grep default: /etc/inittab

id:3:initdefault:

Уровень запуска можно быстро поменять с init

# init 5 # переход с runlevel 3 на 5

# init 6 # reboot

0 # Выключить, poweroff, shutdown now -h

1, S # Single-User mode

2 # Multi-user без сети

3 # Multi-user с сетью

4 # Не используется

5 # Multi-user с X

6 # Перезагрузка, Reboot  
  
  
chkconfig - конфигурирует какие программы будут запускаться при запуске OS с определенным runlevel

# chkconfig --list # Список всех init скриптов

# chkconfig --list sshd # Статус sshd

# chkconfig sshd --level 35 on # Запуск sshd на уровне 3 и 5

# chkconfig sshd off # Отключить sshd для всех runlevels

В Debian и основанных на Debian дистрибутивах используется update-rc.d для управления скриптами runlevels . По умолчанию стартует в 2,3,4,5 и выключается в 0,1 и 6.

# update-rc.d sshd defaults # активирует sshd на runlevels по умолчанию

# update-rc.d sshd start 20 2 3 4 5 . stop 20 0 1 6 . # Применяем аргументы

# update-rc.d -f sshd remove # Выключаем sshd для всех runlevels

# shutdown -h now # Выключаем систему

2.9 Пользователи

# id # Показывает активного пользователя с логином и группой uid и gid

# last # Выводит список последних логинов (подключений) в стистему

# who # Список подключенных пользователей к системе

# groupadd admin # Добовляет группу "admin" и пользователя myname

# useradd -c "Имя Фамилия" -g admin -m myname

# usermod -a -G <group> <user> # Добовляет существующего пользователя в группу (Debian)

# groupmod -A <user> <group> # Добовляет существующего пользователя в группу (SuSE)

# userdel myname # Удаляет пользователя myname (Linux/Solaris)  
  
2.10 Память

Резидентная память - память в оперативке

Анонимная память (anon) - память без привязки к файлу

Page fault - обращение к памяти, trap

# vm.overcommit # параметр чтобы не переиспользовать виртуальную память

# 1 - не следим за оверкоммитом,

# 0-ограничеваем виртуальную память,чтобы не получить намного больше чем есть - спец механизм (но руут не ограничен),

# 2 - возможно выставлять лимиты) cat /proc/meminfo commitLimmit commit\_AS

# numactl --hardware # информация о numa облстях памяти с привязкой к ЦПУ

# cat /proc/zoneinfo # зоны памяти (физ смысл, разновидности и группы страниц памяти)

# vmtouch filename # смотрим сколько у нас файла в кэше

# kswapd # вытеснение из кеша (dirty)

# vm.dirty\_ratio # как часть файла писать на диск

# pmap -x PID # смотрим VMA (виртуальные группы страниц памяти)

# ulimit # меняем для процесса органичения стека

# mmap # отоброжение файла в адресное пространство

# sar -B # -B: статистика по страницам, -r: утилизация памяти, -R: статистика использования памяти

# durty - страницы не сброшеные в память

# readhead # как читать файлы с диска (стратегия по чтению черерез fd)

# blockdev # управление блочным девайсом

Все операции чтения и записи работают через пейджкэшь

pages recliming - освобождение памяти(sync, disareablw, swap, unreclaim)

LRU листы - организация очередей в cache может быть active innactive (meminfo)

page fault minor - выделение без чтения с диска

page fault major - с чтением диска

# echo "-17" > /proc/PID/oom\_adj; # oomkiller, принудительное освобождение памяти (убиваем процесс),

# -17 отключить oomkiller (-16..15) веса, можно посмотреть в oom\_score текущий  
  
3 Процессы

3.1 Просмотр процессов

<PID> - уникальный номер(идентификатор) процесса

# ps -auxefw # Расширенный список всех запущенных процессов

# ps axww | grep cron

586 ?? Is 0:01.48 /usr/sbin/cron -s

# ps axjf # Процессы как дерево процессов

# ps aux | grep 'ss[h]' # Найти все ssh pids без grep pid

# pgrep -l sshd # Найти PIDs процессов по имени (части имени)

# echo $$ # PID нашего shell

# fuser -va 22/tcp # Процессы использующие порт 22

# pmap PID # Карта памяти процесса (поиск утечки памяти), используемые библиотеки

# fuser -va /home # Процессы работаюшие с разделом /home

3.2 Приоритеты

Изменить приоритет запущенного процесса можно с renice. Отрицательные числа имеют наивысший приоритет. Границы от -20 до 20

# renice -5 586 # Приоритетный

586: old priority 0, new priority -5

Приоритет запускаемых процессов устанавливаем с nice.

# nice -n -5 top # Высокий приоритет (/bin/nice)

# nice -n 5 top # Низкий приоритет (/bin/nice)

# nice +5 top # Низкий приоритет

nice меняет планировщик CPU, ionice меняет планировщик I/O дисков.

# ionice c3 -p123 # Устанавливает класс idle для pid 123

# ionice -c2 -n0 firefox # Запускает firefox с best effort и высоким приоритетом

# ionice -c3 -p$$ # Устанавливает актуальный shell idle приоритет

Последняя команда удобна для отладки и компиляции больших проектов. Кождая команда запущенная из этого shell будет иметь подобный приоритет (приоритет $$, если $$ PID shell).

3.3 Background/Foreground

Процесс запущенный в shell может быть убран на задний план(Background), а потом вызван на передний план(Foregroun) нажатием <Ctrl>+<Z>. Запускайте процесс с nohup, чтоб он продолжил работу, когда shell будет закрыт.

# ping ya.ru > ping\_ya.ru.log

<Ctrl>+<Z> # ping приостановлен, <Ctrl>+<Z>

# bg # убрали в background и продолжили выполнение

# jobs -l # смотрим что в background

[1] - 31574 Running ping ya.ru > ping\_ya.ru.log

[2] + 36233 Suspended (tty output) top

# fg %2 # Вызвать процесс в foreground

# nohup ping -i 60 > ping\_ya.ru.log &

3.4 Top, htop

Top - выводит информацию в реальном времени о запущенных процессах. Htop - продвинутая версия top, ставится из репозиториев.

<u> [username] Показывает процессы принадлежащие пользователю username

<k> [pid] Убить процесс с pid

<1> Покажет статистику использования процессоров

<R> Сортировка

3.5 Kill, signals

# ping -i 60 ya.ru > ping\_ya.ru.log &

[1] 4712

# kill -s TERM 4712 # также kill -15 4712

# killall -1 httpd # Завершить HUP процесс по имени

# pkill -9 http # Завершить TERM процесс по имени (части имени)

# pkill -TERM -u www # Завершить TERM процессы принадлежащие www

# fuser -k -TERM -m /home # Завершить все процессы использующие /home (для umount)

Сигналы:

# -1 HUP # Дать отбой, перегрузка конфигурационных файлов и перезапуск программы

# -2 INT # Прервать

# -3 QUIT # Выйти

# -9 KILL # Прекратить все выполняющиеся действия и завершить работу

# -15 TERM # Мягкое завершение с удаление порожденных процессов и закрытием файлов  
  
4 Файловая система  
Права

1 --x execute/выполнять | Mode 764 = exec/read/write | read/write | read

2 -w- write/писать | For: |-- Owner --| |- Group-| |Oth|

3 r-- read/читать |

ugo=a u=user, g=group, o=others, a=everyone

# chmod [OPTION] MODE[,MODE] FILE # MODE имеет вид [ugoa]\*([-+=]([rwxXst]))

# chmod 640 /var/log/maillog # Установили log -rw-r-----

# chmod u=rw,g=r,o= /var/log/maillog # Установили log -rw-r-----

# chmod -R o-r /home/\* # Рекурсивно запрещаем чтение всем пользователям в /home/\*

# chmod u+s /path/to/prog # Устанавливаем SUID bit

# find / -perm -u+s -print # Находим все программы использующие SUID bit

# chown user:group /path/to/file # Меняем хозяина и группу файла

# chgrp group /path/to/file # Меняем группу файла

# chmod 640 `find ./ -type f -print` # Меняем права на 640 для всех файлов

# chmod 751 `find ./ -type d -print` # Меняем права на 751 для всех директорий

# umask 0174 /test

# touch /test/foo

# mkdir /test/dir

# ls -la /test

drw-----wx 2 root root 512 Sep 1 20:59 dir

-rw-----w- 1 root root 0 Sep 1 20:59 foo

4.2 Информация о дисках

# hdparm -I /dev/sda # Информация о IDE/ATA (Linux)

# fdisk /dev/ad2 # Работа с таблицей разделов

# smartctl -a /dev/ad2 # Отобразить информацию SMART

4.3 Точки монтиования, разделы, использование дисков

# mount | column -t # Смонтированные файловые системы

# df # Свободное пространство на дисках

# cat /proc/partitions # Все зарегетсрированные разделы (Linux)

# du -sh \* # Список с размерами папок

# du -csh # Размер текущей папки (сумма поддиректорий)

# du -ks \* | sort -n -r # Размер в kilobytes с сортировкой

# ls -lSr # Показать файлы, большие в конце  
  
5 Сеть  
5.1 Устранение ошибок (+ анализ трафика)

# arping 192.168.1.2 # Ping на канальном уровне (ethernet layer)

# tcptraceroute -f 5 zabrosov.ru # Используем tcp вместо icmp для трассировки через firewalls

# ping zabrosov.ru

# traceroute zabrosov.ru # Маршрут до zabrosov.ru

# netstat -s # Сетевая статистика для каждого протокола системы  
  
5.5 Занятые порты

# netstat -an | grep LISTEN

# lsof -i # Просмотр всех соединений

# socklist # Список открытых сокетов

# netstat -anp --udp --tcp | grep LISTEN

# netstat -tup # Активные соединения

# netstat -tupl # Список открытых портов системы

5.6 Firewall

# iptables -L -n -v # Листинг правила

Open the iptables firewall

# iptables -P INPUT ACCEPT # Разрешить все

# iptables -P FORWARD ACCEPT

# iptables -P OUTPUT ACCEPT

# iptables -Z # Обнулить все счетчики во всех цепочках

# iptables -F # Очистить все цепочки

# iptables -X # Удалить все цепочки  
  
5.9 DNS

Настройки DNS храняться в /etc/resolve.conf:

nameserver 8.8.8.8

search mydomain.local studdomain.lab

domain mydomain.local

# hostname -d # Имя системы ( аналог: dnsdomainname)

dig - утилита для тестирования DNS. В качестве DNS сервера используем свободный google dns с ip: 8.8.8.8

# dig zabrosov.ru

;; ANSWER SECTION:

zabrosov.ru. 600 IN A 77.221.130.2

;; SERVER: 8.8.8.8#53(8.8.8.8)

Некоторые полезные запросы:

# dig MX zabrosov.ru

# dig @127.0.0.1 NS zabrosov.ru # Тест локального DNS-сервера (если есть)

# dig @8.8.8.8 NS MX zabrosov.ru # Запрос к внешнему DNS-сервер

# dig AXFR @ns1.infobox.org zabrosov.ru # Получить зону (zone transfer)

# аналог dig

# host -t MX zabrosov.ru # Посмотреть MX записи

# host -t NS -T zabrosov.ru # Получить NS записи через tcp

# host -a zabrosov.ru # Вся информция по домену

Имя по IP адресу (in-addr.arpa. , обратная зона). Используется dig, host and nslookup:

# dig -x 77.221.130.2

# host 77.221.130.2

# nslookup 77.221.130.2

Для локального преобразовния имен (в рамках системы) можно использовть файл /etc/hosts:

# cat /etc/hosts

127.0.0.1 myibm localhost.localdomain localhost

192.168.1.2 myibm.zabrosov.local myibm

5.10 DHCP

Интерфейс используемый по умолчанию - eth0

# dhcpcd -n eth0 # Обновить (не всегда работает)

# dhcpcd -k eth0 # Освободить

Полная информация о аренде:

/var/lib/dhclient/dhclient-eth0.leases  
  
8.2 tar

tar - архивирование файлов и директорий. Сам по себе tar, это не сжатый архив, сжатые архивы имеют расширения .tgz или .tar.gz (gzip) или .tbz (bzip2).

Создание архива tar:

# cd /

# tar -cf home.tar home/ # Создать архив, поместив в него директорию /home ( ключ -c, для создания )

# tar -czf home.tgz home/ # То-же, но с gzip компрессией

# tar -cjf home.tbz home/ # То-же, но с bzip2 компрессией

# Рекомендуется использовть относительные пути, чтоб можно было распаковать в любое место

# tar -C /usr -czf local.tgz local/etc local/www # Cоздаlbv архив, содержащий директории /usr/local/etc,

# /usr/local/www, директория local/ должна быть началом дерева

# tar -C /usr -xzf local.tgz # Распаковать архив директорию local в дерево /usr

# cd /usr; tar -xzf local.tgz # То-же, что выше

Распаковать архив tar:

# tar -tzf home.tgz # Просмотр содержимого архива без его распаковки (листинг)

# tar -xf home.tar # Распаковать архив в текущую папку(ключ "x" для распаковки)

# tar -xzf home.tgz # То-же для архива с zip компрессией

# tar -xjf home.tbz # То-же для архива с bzip2 компрессией

# tar -xjf home.tbz home/zabrosov.ru/file.txt # Распаковать один файл

Полезно:

# tar c dir/ | gzip | ssh user@remote 'dd of=dir.tgz' # Создать архив, содержащий директорию dir/ и сохранить удаленно

# tar cvf - `find . -print` > backup.tar # Создать архив с текущей директорией

# tar -cf - -C /etc . | tar xpf - -C /backup/etc # Копировать директории

# tar -cf - -C /etc . | ssh user@remote tar xpf - -C /backup/etc # Удаленное копирование

# tar -czf home.tgz --exclude '\*.o' --exclude 'tmp/' home/ # Создать архив с сжатием, исключив '\*.o', tmp/  
  
  
11.2 screen

screen - оконный менеджер виртуальных терминалов

screen имеет две основные функции:

- Запуск нескольких сессий терминала, в одном окне.

- Запуск программ отдельно от терминала в фоновом режиме. Терминал может быть отключен и переподключен позже.

# screen # Запустили менеджер

# tail -f /var/log/messages # Вывод логов для примера листинга

# Отсоединим screen от физического терминала нажав: <Ctrl>+<a> <Ctrl>+<d>

# screen -rd # Переподключиться

# screen -x # Подключиться в многоэкранном режиме (полезно для совместной работы)

# echo "defscrollback 5000" > ~/.screenrc # Увеличить буфер до 5000 (по умолчанию 100)

# Все команды screen начинаются с <Ctrl>+<a>

<Ctrl>+<a> <?> # Справка и список доступных функций

<Ctrl>+<a> <c> # Создать новое окно (терминал)

<Ctrl>+<a> <Ctrl>+<n> и <Ctrl>+<a> <Ctrl>+<p> # Переключиться на предыдущий или следующий экран

<Ctrl>+<a> <Ctrl+<N> # Где N, число от 0 до 9, что-бы переключится на окно с соответствующим номером

<Ctrl>+<a> <"> # Получить список запущенных окон

<Ctrl>+<a> <a> # Очистить пропущенный <Ctrl>+<a>

<Ctrl>+<a> <Ctrl>+<d> # Отключиться, оставив сессию запущенной в фоновом режиме

<Ctrl>+<a> <x> # Заблокировать терминал паролем

<Ctrl>+<a> <[> # запуск режима scrollback mode, для выхода <esc>

<C>+<u> # Прокрутка на полстраницы вверх

<C>+<b> # Прокрутка на страницу вверх

<C>+<d> # Прокрутка на полстраницы вниз

<C>+<f> # Прокрутка на страницу вниз

</> # Поиск вперед

<?> # Поиск назад

Сессия терминала прерывается, когда будет закрыта работающая программа и сделан выход с терминала.

11.3 find

Некоторые опции:

-xdev # Оставаться на то-же файловой системе

-exec cmd {} \; # Выполнить команду, если есть {}, то find заменяет их на путь и имя файла найденного файла

-iname # То-же, что и -name (поиск по имени), но без учета регистр

-ls # Показать информацию о файле (как ls -la)

-size n # Размер в блоках или байтах, n (равно n блоков), +n (более n блоков), -n (менее n блоков),

# доступные обозначения размеров: k, M, G, T, P

-cmin n # Статус файла был изменен N минут назад

# find . -type f ! -perm -444 # Найти невидимые для всех файлы

# find . -type d ! -perm -111 # Поиск недоступных для всех папок

# find /home/user/ -cmin 10 -print # Файлы созданные или модифицированные за последние 10 минут

# find . -name '\*.[ch]' | xargs grep -E 'expr' # Найти 'expr' в текущей директории

# find / -name "\*.core" | xargs rm # Найти и удалить аварийные дампы(так-же можно искать core.\*)

# find / -name "\*.core" -print -exec rm {} \; # Другой сиснтаксис

# Найти все графические файлы и создать архив, iname -регистронезависимо. -r -добавить

# find . \( -iname "\*.png" -o -iname "\*.jpg" \) -print -exec tar -rf images.tar {} \;

# find . -type f -name "\*.txt" ! -name README.txt -print # Исключая файлы README.txt

# find /var/ -size +10M -exec ls -lh {} \; # Найти файлы больше 10MB

# find /var/ -size +10M -ls # То-же, что и выше

# find . -size +10M -size -50M -print # Найти файлы больше 10MB, но меньше 50MB

# Найти файлы, принадлежащие определенному пользователю и с определенными правами

# find / -type f -user root -perm -4000 -exec ls -l {} \;

Будьте осторожны при использовании xargs или exec, они могут возвращать неверный результат если имена файлов или директорий содержат пробелы. Используйте -print0 | xargs -0, вместо | xargs. Опция -print0 должна быть последней.

# find . -type f | xargs ls -l # Не будет работать при наличии пробелов в именах

# find . -type f -print0 | xargs -0 ls -l # Будет работать нормально с пробелами

# find . -type f -exec ls -l '{}' \; # Или используйте с -exec, ковычки '{}'  
  
12 Shell

В большинстве Linux, в качестве системной оболочки, используется bash.

# cmd 1> file # Пренаправить stdout (стандартный вывод) в файл

# cmd 2> file # Перенаправить stderr (стандартный вывод ошибок ) в файл

# cmd 1>> file # Перенаправить stdout и дописать его в файл

# cmd &> file # Перенаправить все stdout и stderr в файл

# cmd >file 2>&1 # Пренаправить stderr в stdout и потом в файл

# cmd1 | cmd2 # Передать stdout на вход команды cmd2

# cmd1 2> &1 | cmd2 # Передать stdout и stderr команде cmd2

Настройка оболочки в файле конфигурации ~/.bashrc ( так-же может быть ~/.bash\_profile )

# ~/.inputrc

bind '"\e[A"':history-search-backward # Использовать клавиши "вверх" и "вниз" для поиска

bind '"\e[B"':history-search-forward # История введенных команд

# .bashrc

set -o emacs # Установить emacs режим в bash (см. ниже)

set bell-style visible # Не подавать звуковой сигна, инверировать цвета

[user@host]/path/todir> # Настройка строки приглашения shell

PS1="\[\033[1;30m\][\[\033[1;34m\]\u\[\033[1;30m\]"

PS1="$PS1@\[\033[0;33m\]\h\[\033[1;30m\]]\[\033[0;37m\]"

PS1="$PS1\w\[\033[1;30m\]>\[\033[0m\]"

export PS1='\033[00;32m\][`date +%d" "%h" "%Y" "%T`] \u@\h \w\n \$\[\033[00m\] ' # видно юзера/хост, дату и время

export HISTTIMEFORMAT='%F %T ' # Ведение лога истории с datestamp'ом

PROMPT\_COMMAND='history -a;history -n' # Занесение команды в .bash\_history сразу же,

# после нажатия enter, удобно при работе в нескольких сессиях

export HISTSIZE=100500 # Увеличение размера хистори

export HISTCONTROL=ignoredups # не заносить в хистори повторяющиеся друг за другом команды

export HISTIGNORE="&:ls:[bf]g:exit:[ ]\*:ssh:history" # Отключаем занесение «бесполезных» с точки зрения истории команд

export EDITOR=nano # nano станет редактором по умолчанию

alias u='sudo pacman -Syu' # Используем сокращения для популярных комманд

<Ctrl>+<R> — вводишь слово, с которым надо найти команду в истории. Повторные <Ctrl>+<R> перебирают все команды в истории с этим словом.

<Ctrl>+<L> — очистка экрана.

<Ctrl>+<A> — замена <Home>, <Ctrl>+<E> — <End>

<Ctrl>+<U> — удаление текста от курсора до начала строки, <Ctrl>+<K> — удаление от курсора до конца строки, <Ctrl>+<Y> — вставить удаленный предыдущими примерами кусок текста, <Ctrl>+<C> — удаление всей строки (обычно работает как отмена команды).

<Ctrl>+<T> — меняет местами символ под курсором и предыдущий набранный (при опечатке вида /dev/dsa вместо /dev/sda)

^^ — После попытки выполнить команду, набранную с ошибкой, строка вида ^ошибка^правильно запустит на выполнение исправленную команду.

# — если вместо набранной команды надо сначала выполнить другую, можно закомментировать строку и нажать <enter>, впоследствии вернуться к команде в истории, раскомментировать и выполнить.

<Alt>+<.> — подставляет к текущей команде аргумент предыдущей.  
  
  
16 SSL сертификаты

SSL - Secure Socket Layer, криптографический протокол, использующий шифрование открытым ключем, для защиты передаваемых по сети данных. Протокол SSL, является важным элементом политики безопасности системы. SSL сертификат - электронный документ, используемый для подтверждения принадлежности транзакции тому или иному серверу и установления защищенного соединения между клиентом и сервером с шифрованием трафика. Часто используется на защищенных Веб серверах (https) или Mail серверах (imaps)

- Клиент должен создать сертификат, со всеми необходимыми данными

- Отправить запрос на сертификацию в один из "центров сертификации" (CA). Так-же на данном этапе, будет создан приватный ключ на локальной машине

- После обработки запроса, сертификат подписывается секретным ключем CA. Клиент имея публичный ключ CA, проверяет подлинность сертификата и далее может использовать его

- Если необходимо, можно объединить сертификат и ключ в один файл

Конфигурация OpenSSL

В данном примере мы будем использовать директорию /usr/local/certs. Проверьте и отредактируйте файл /etc/pki/tls/openssl.cnf, согласно вашей конфигурации.

#nano /etc/pki/tls/openssl.cnf

[ CA\_default ]

dir = /etc/pki/CA # Храним все тут

certs = $dir/certs # Где хранить сертификаты

crl\_dir = $dir/crl # Где хранить списки отзыва сертификатов (CRL)

database = $dir/index.txt # Индексный файл базы данных

Создать сертификат полномочий

Если у нас нет сертификата, подписанного CA, и вы не планируете отправлять запрос на сертификацию, можно создать свой сертификат

# openssl req -new -x509 -days 730 -config /etc/pki/tls/openssl.cnf -keyout CA/private/cakey.pem -out CA/cacert.pem  
  
  
Запрос сертификации (CSR)

# openssl req -new -keyout newkey.pem -out newreq.pem \

-config /etc/ssl/openssl.cnf

# openssl req -nodes -new -keyout newkey.pem -out newreq.pem \

-config /etc/ssl/openssl.cnf # Без шифрования ключа

Сохраним запрос (newreq.pem), он может быть отправлен снова, для следующего обновления, подпись ограничивает срок действия сертификата. Кроме того, в процессе, будет создан приватный ключ newkey.pem

Подпись сертификата

Подписанный CA сертификат является действующим.

# cat newreq.pem newkey.pem > new.pem # Заменим "servername" на имя своего сервера

# openssl ca -policy policy\_anything -out servernamecert.pem -config /etc/ssl/openssl.cnf -infiles new.pem

# mv newkey.pem servernamekey.pem

Теперь servernamekey.pem - содержит приватный ключ а servernamecert.pem - сертификат сервера.

Создание объединенного сертификата

IMAP серверу нужно иметь все приватные ключи и серверные сертификаты в одном файле, файл должен храниться в безопасном месте.

Создадим файл servername.pem содержащий и сертификаты и ключи:

1.Открыть файл servernamekey.pem в текстовом редакторе и скопировать приватный ключ в файл servername.pem

2.Открыть файл servernamecert.pem в текстовом редакторе и скопировать сертификат в файл servername.pem

# nano /etc/pki/CA/certs/servername.pem

-----BEGIN RSA PRIVATE KEY-----

MIICXQIBAAKBgQDutWy+o/XZ/[...]qK5LqQgT3c9dU6fcR+WuSs6aejdEDDqBRQ

-----END RSA PRIVATE KEY-----

-----BEGIN CERTIFICATE-----

MIIERzCCA7CgAwIBAgIBBDANB[...]iG9w0BAQQFADCBxTELMAkGA1UEBhMCREUx

-----END CERTIFICATE-----  
  
  
# Итог /etc/pki/

CA/private/cakey.pem (CA server private key)  
CA/cacert.pem (CA server public key)

certs/servernamekey.pem (server private key)

certs/servernamecert.pem (server signed certificate)

certs/servername.pem (server certificate with private key)

Информация о сертификате

# openssl x509 -text -in servernamecert.pem # Посмотр информации о сертификате

# openssl req -noout -text -in server.csr # Информация запроса

# openssl s\_client -connect zabrosov.ru:443 # Проверить сертификат Веб-сервера