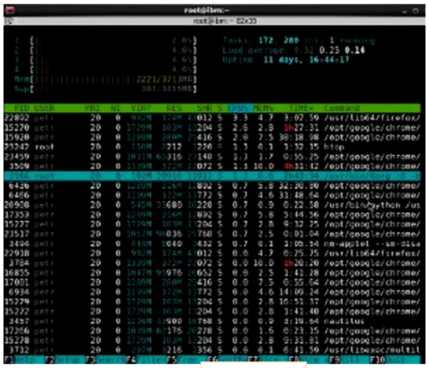
ZABROSOV Команды Linux | Базы данных | Все вместе | O! 16



Сайт справочник - неверное определение для www.zabrosov.ru, скорее всего - это, сайт подсказка, заметка и шпаргалка.
Трудно удержать в памяти большой объем информации, если каждый день её (информацию) не используешь.
Приходится вести заметки и писать напоминалки.

Тут можно найти информацию о основных командах Linux с примерами и комментариями для повседневной жизни в консоли (у некоторых эта жизнь очень насыщена ;)).

Заметки ориентированы на дистрибутивы: Centos (RedHat), Fedora, Debian, Ubuntu (именно в таком порядке).
Но будет актуален практически во всех UNIX системах.

Текущая версия: 29-3-15

С удовольствием приму от вас уточнения, исправления, tools & hacks на: **zabrosov[at]gmail.com**

Справочник основных команд Linux с примерами

Поделиться...

1 Общая информация

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)+(см. regexp)

```
# 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 релактирование файла
```

1.5 nano

nano - редактор файлов, более дружелюбный чем vim ;)

```
# nano /etc/hosts                         # просмотр и релактирование 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

nano-памятка

2 Система

2.1 Загрузка | 2.2 Hardware | 2.3 Ресурсы и статистика | 2.4 Ограничения | 2.5 Runlevels | 2.6 Восстановить пароль root | 2.7 kernel | 2.8 grub | 2.9 Пользователи | 2.10 Память

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
```

2.6 Восстановить пароль root

Вариант 1
Когда стартует grub, появляется возможность выбора режима загрузки в grub_boot_screen (иногда при загрузке нужно нажать F4)
1.Нажимем <*пробел*> - попадаем в меню выбора вариантов загрузки kernel и initrd.
2.Нажимаем <*e*> - для редактирования режима загрузки и дописывем S (загрузка в single user mode).

```
kernel /vmlinuz-2.6.35.11-83.fc14.i686 ro root=/dev/vg00/logvol00 rhgb quiet S
```

3.Нажимаем <*b*> для загрузки с установленным параметром.
4.Выполняем команду и вводим нужный пароль.

```
#passwd
Смена пароля для пользователя root.
Новый пароль :
```

Вариант 2
В момент загрузки grub, задаем:

```
init=/bin/sh
```

Ядро примонтирует корневой раздел, init запустит shell. Выполняем:

```
# mount -o remount,rw /
# passwd                            # или удаляем пароль root в (/etc/shadow)
# sync; mount -o remount,ro /       # sync до перемонтирования в read only
# reboot
```

Вариант 3
1.Загружаемся с внешнего носителя в rescue mode (liveCD или installation CD).
2.Находим корневой раздел с помошью fdisk.
3.Выполняем:

```
# mount -o rw /dev/ad4s3a /mnt      # монтируем корневой раздел в /mnt системы с LiveCD
# chroot /mnt                       # chroot в /mnt
# passwd
# reboot
```

2.7 kernel (update kernel)

Варианты ядра для x86:
1.Regular: несколько процессоров, максимально 4GB RAM
2.PAE: 32 процессора, 16GB RAM
3.Xen: поддержка виртуализации
Варианты ядра для x86_64:
1.Regular: 64 процессора, максимально 256GB RAM
2.Xen: поддержка виртуализации
Ядро обычно устанавливается в /boot/vmlinuz-*

```
# /proc/<PID>                       # информация о процессах (top,ps)
# /proc/cmdline                     # boot time опции
# /proc/cpuinfo                     # CPU
# /proc/mdstat                      # software RAID (mdadm)
# /proc/meminfo                     # использование памяти (free, vmstat)
# /proc/swaps                       # swap
# /proc/modules                     # загруженные модули (lsmod)
# /proc/mounts                      # смонтированные файловые системы(mounts)
# /proc/net                         # сетевая активность и конфигурация (ifconfig,netstat)
# /proc/partitions                  # block devices
# /proc/version                     # версия Linux kernel (uname)
```

Некоторые устройства:

```
# block devices
# /dev/hda, /dev/hdc                    # IDE hdd, CD/DVD-ROM
# /dev/sda, /dev/sdb                    # SCSI, SATA, USB-hdd
# /dev/md0, /dev/md1                    # Software RAID
# character devices
# /dev/tty[0-6]                         # virtual consoles
# /dev/null, /dev/zero                  # software devices, "ноль"
```

```
# /dev/random, /dev/urandome                # генератор случайных чисел
```

2.8 grub

```
# /sbin/grub-install /dev/hda               # обновить MBR
# grub-crypt                                # получить hash sha-512, для пароля на grub
# nano /boot/grub/grub.conf
password --sha-512 $6$tdi8VPVCnSkGZbjw$oCy/ # вставляем строчку с hash
default=0
```

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)
```

В /etc/shadow хранится пара: hash пароля и логин.

```
# for USER in petr alex vadim sergey      # Устанавливаем перечисленным пользователям пароль: password
# do
#   useradd $USER
#   echo password | passwd --stdin $USER
# done

# while read u n      # Добовляем пользователей с паролем: password
#  do
#    useradd --comment "$n" --create-home $u
#    echo password | passwd --stdin $u
#  done
```

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 | 3.2 Приоритеты | 3.3 Background/Foreground | 3.4 top | 3.5 kill

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 работающие 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

4.1 Права | 4.2 Информация о дисках | 4.3 Использование дисков | 4.4 Открытые файлы | 4.5 Mount/remount | 4.6 Увеличить SWAP | 4.7 Mount SMB | 4.8 Монтируем образ | 4.9 Запись ISO | 4.10 Создать образ | 4.11 Memory disk | 4.12 Производительность дисков

Права

```
1 --x execute/выполнять       |       Mode 764 = exec/read/write | read/write | read
2 -w- write/писать            |       For:       |--  Owner  --|   |- Group-|   |Oth|
4 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
```

4.4 Блокировка файлов, кто использует файлы?

```
# umount /home/
umount: unmount of /home              # Выполнить umount не возможно, файлы в home заблокированы
   failed: Device busy
```

Найти открытые файлы, для Xorg:

```
# ps ax | grep Xorg | awk '{print $1}'
1252
# fstat -p 1252
USER     CMD          PID   FD MOUNT      INUM MODE         SZ|DV R/W
root     Xorg        1252 root /             2 drwxr-xr-x     512  r
root     Xorg        1252 text /usr     216016 -rws--x--x  1679848 r
root     Xorg        1252    0 /var     212042 -rw-r--r--   56987  w
```

Файл с inum 212042 нашелся в /var:

```
# find -x /var -inum 212042
/var/log/Xorg.0.log
```

Поиск открытых файлов в точках монтирования с *fuser* или *lsof*

```
# fuser -m /home                      # Список процессов использующих /home
# lsof /home
COMMAND   PID    USER   FD   TYPE DEVICE    SIZE     NODE NAME
tcsh    29029 eedcoba  cwd    DIR   0,18   12288  1048587 /home/eedcoba (guam:/home)
lsof    29140 eedcoba  cwd    DIR   0,18   12288  1048587 /home/eedcoba (guam:/home)
```

Приложение:

```
ps ax | grep Xorg | awk '{print $1}'
3324
# lsof -p 3324
COMMAND   PID    USER   FD   TYPE DEVICE    SIZE    NODE NAME
Xorg    3324 root    0w   REG        8,6   56296      12492 /var/log/Xorg.0.log
```

Файл:

```
# lsof /var/log/Xorg.0.log
COMMAND  PID USER   FD   TYPE DEVICE  SIZE  NODE NAME
Xorg    3324 root    0w   REG    8,6 56296 12492 /var/log/Xorg.0.log
```

4.5 Mount/remount

```
# mount /cdrom                           #если есть в /etc/fstab
/dev/cdrom   /media/cdrom  subfs noauto,fs=cdfss,ro,procuid,nosuid,nodev,exec 0 0  #строчка из /etc/fstab
# mount -t auto /dev/cdrom /mnt/cdrom    # монтруем CD-ROM
# mount /dev/hdc -t iso9660 -r /cdrom    # IDE
# mount /dev/scd0 -t iso9660 -r /cdrom   # SCSI CD-ROM
# mount /dev/sdc0 -t ntfs-3g /windows    # SCSI

# mount -o remount,ro /                  # перемонтирование без размонтирования
# mount -n -o remount,rw /               # перемонтировать в режиме чтения|записи
```

4.6 Увеличиваем SWAP

Иногда нужно увеличить swap, прямо сейчас, на лету

```
# dd if=/dev/zero of=/swap2gb bs=1024k count=2000
# mkswap /swap2gb                     # создаем место для swap
# swapon /swap2gb                     # запускаем swap. сейчас используется swap
# swapoff /swap2gb                    # отключаем swap
# rm /swap2gb
```

4.7 Монтируем SMB share

cifs использует ip или DNS имя

```
# smbclient -U user -I 192.168.1.2 -L //smbshare/    # Смотрим шары
# mount -t smbfs -o username=winuser //smbserver/myshare /mnt/smbshare
# mount -t cifs -o username=winuser,password=winpwd //192.168.1.2/myshare /mnt/share
```

Если используется mount.cifs, то можно хранить учетные данные в файле /home/user /.smb:

```
username=winuser
```

```
password=winpwd
```

Монтируем:

```
# mount -t cifs -o credentials=/home/user/.smb //192.168.1.2/myshare /mnt/smbshare
```

4.8 Монтируем image (образ)

```
# mount -t iso9660 -o loop file.iso /mnt               # Монтируем CD-image
# mount -t ext3 -o loop file.img /mnt                  # Монтируем образ с ext3 fs
```

4.9 Создание и запись ISO образа

Без *conv=notrunc* содержимое будет меньше объема диска, если данные не занимают весь диск (DVD запишем на CD).

```
# dd if=/dev/hdc of=/tmp/mycd.iso bs=2048 conv=notrunc
```

Используем *mkisofs* для создания CD/DVD образа из файлов в папке. Для переодоления ограничений на имена файлов: -r разрешает Rock Ridge расширения для UNIX истем, -J разрешает Joliet расширения используемые Microsoft ОС. -L разрешает имена согласно ISO9660.

```
# mkisofs -J -L -r -V TITLE -o imagefile.iso /path/to/dir
```

Запись:

```
# cdrecord -scanbus                  # Находим устройство для записи (1,0,0)
# cdrecord dev=1,0,0 imagefile.iso

# cdrecord dev=ATAPI -scanbus        # Можно использовать native ATAPI
```

Конвертируем Nero .nrg файл в .iso

```
# dd bs=1k if=imagefile.nrg of=imagefile.iso skip=300  # Nero просто добавляет 300Kb заголовок к нормальному iso
```

4.10 Монтируем образ (файл как образ)

```
# dd if=/dev/zero of=/usr/vdisk.img bs=1024k count=1024
# mkfs.ext3 /usr/vdisk.img
# mount -o loop /usr/vdisk.img /mnt
# umount /mnt; rm /usr/vdisk.img                       # Очистили
```

/dev/zero быстрее *urandom*, но менее безопасный.

```
# dd if=/dev/urandom of=/usr/vdisk.img bs=1024k count=1024
# losetup /dev/loop0 /usr/vdisk.img                    # Создали и определили /dev/loop0
# mkfs.ext3 /dev/loop0
# mount /dev/loop0 /mnt
# losetup -a                                           # Проверка использования loops
# umount /mnt
# losetup -d /dev/loop0                                # Удаление
# rm /usr/vdisk.img
```

4.11 Memdisk

Tmpfs очень быстрая файловая система для IO приложений. Создадим 64 MB раздел, смонтируемый в /memdisk:

```
# mount -t tmpfs -osize=64m tmpfs /memdisk
```

4.12 Производительность дисков

```
# time dd if=/dev/ad4s3c of=/dev/null bs=1024k count=1000     # Читаем и пишим 1GB в /home (/dev/ad4s3c)
# time dd if=/dev/zero bs=1024k count=1000 of=/home/1Gb.file
# hdparm -tT /dev/hda
```

5 Сеть

5.1 Устранение ошибок | 5.2 Routing | 5.3 Дополнительный IP | 5.4 Сменить MAC | 5.5 Порты | 5.6 Firewall | 5.7 Разрешить роутинг | 5.8 NAT | 5.9 DNS | 5.10 DHCP | 5.11 Анализ трафика | 5.12 QoS | 5.13 NIS | 5.14 Netcat (nc)

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
```

Работа с NIC

```
# ethtool eth0              # Смотрим свойства nic
# ethtool -s eth0 speed 100 duplex full # Устанавливаем 100Mbit Full duplex
# ethtool -s eth0 autoneg off # Отключаем auto negotiation
# ethtool -p eth1           # Мигнуть светодиодом (если поддерживается)
# ip link show              # Показать все интерфейсы (или ifconfig)
# ip link set eth0 up       # Включить eth0 (или отключить - down)
# ifconfig eth0 up          # Включить eth0 (или отключить - down)
# ip addr show              # Посмотреть все IP (аналог ifconfig)
# ip neigh show             # arp
# arp -a                    # Работа с arp
```

eth0 из shell

```
# Вывод списка NIC с сортировой
# ifconfig | sed 's/  / /g' | cut -d" " -f1 | uniq | grep -E "[a-z0-9]+" | sort -r
# Вывод всех IP машины с сортировой
# ifconfig | sed '/.*inet addr:/!d;s///;s/ .*//'|sort -t. -k1,1n -k2,2n -k3,3n -k4,4n

# nano /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=none
BROADCAST=192.168.1.255
HWADDR=00:C0:26:30:EA:32
IPADDR=192.168.1.2
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
GATEWAY=192.168.1.1
TYPE=Ethernet
USERCTL=no
IPV6INIT=no
PEERDNS=yes
```

5.2 Routing

```
# route -n
# netstat -rn
# ip route
```

Добавить/удалить маршрут

```
# route add -net 192.168.20.0 netmask 255.255.255.0 gw 192.168.1.2
# ip route add 192.168.20.0/24 via 192.168.1.2       # Аналог
# route add -net 192.168.20.0 netmask 255.255.255.0 dev eth0
# route add default gw 192.168.1.2
# ip route add default via 192.168.1.2 dev eth0      # Аналог
# route delete -net 192.168.20.0 netmask 255.255.255.0
```

5.3 Дополнительный IP

```
# ifconfig eth0 192.168.50.254 netmask 255.255.255.0       # Первый IP
# ifconfig eth0:0 192.168.51.254 netmask 255.255.255.0     # Второй IP
# ip addr add 192.168.50.254/24 dev eth0                   # Аналог
# ip addr add 192.168.51.254/24 dev eth0 label eth0:1
```

5.4 Сменить MAC

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:01:02:03:04:05
```

Wireless:

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
```

5.7 Разрешить роутинг

```
# cat /proc/sys/net/ipv4/ip_forward  # Проверка IP forward 0=off, 1=on
# echo 1 > /proc/sys/net/ipv4/ip_forward

или редактируем /etc/sysctl.conf with:
net.ipv4.ip_forward = 1
```

5.8 NAT

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE  # NAT
# iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 20022 -j DNAT \
--to 192.168.1.44:22                                    # Проброс 20022 порта  на внутренний IP порт ssh
# iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 993:995 -j DNAT \
--to 192.168.16.254:993-995                             # Проброс портов 993-995
# iptables -L -t nat                                    # Состояние NAT
# netstat-nat -n                                        # смотрим сесси с IPS
```

Проброс портов можно отменить, заменив в правиле -A на -D.

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
```

5.11 Анализ трафика

Bmon - консольный монитор загрузки сетевых интерфейсов.

Sniff с tcpdump

```
# tcpdump -nl -i eth0 not port ssh and src \(192.168.1.10 or 192.168.1.15\)
# tcpdump -n -i eth0 net 192.168.1.15             # трафик с/на IP
# tcpdump -n -i eth0 net 192.168.1.0/24           # трафик с/в сеть
# tcpdump -l > dump && tail -f dump               # Вывод с записью в файл
# tcpdump -i eth0 -w traffic.eth0                 # Информация о трафике записывается в бинарный файл traffic.eth0
# tcpdump -i eth0 -s 0 -w traffic.eth0            # Запись + загрузка в бинарник
# tcpdump -r traffic.eth0                         # Читаем из файла
# tcpdump port 80                                 # Весь трафик на 80 порт и ответы
# tcpdump host google.com                         # Весь с/на google.com
# tcpdump -i eth0 -X port \(110 or 143\)          # Проверка pop и imap на безопасность
# tcpdump -n -i eth0 icmp                         # Ловим pings
# tcpdump -i eth0 -s 0 -A port 80 | grep GET      # -s 0 весь пакет, -A для ASCII
```

Важные опции:
-A Отображает каждый пакет в открытом виде (текст), без заголовка
-X Показывает пакеты в hex и ASCII
-l Вывод в буффер
-D Печать всех доступных интерфейсов (# tcpdump -D)

Nmap

nmap - сканер портов с возможностью определения OS.

```
# nmap zabrosov.ru              # сканирует все зарезервированные TCP порты на хосте
# nmap -sP 192.168.1.0/24       # Узнать занятые IP в сети 1.0/24 (IP, MAC, hostname)
# nmap -sS -sV -O cb.vu         # Скрытое SYN сканирование с определением типа и версии ОС

# nmap -sX -p 22 192.168.10.10-250 -oN /data/host.txt  # Сканирование машин с ip от 10 по 250 \
                                                       # на порт ssh с записью в файл
# grep  192.168.10 /data/host.txt > /data/host         # Пишем строки с ip адресами в файл
# sed 's/Interesting ports on //g' /data/host | sed 's/://g' |sed 'w /data/host.txt'
# Итог: получаем файл с IP машин, у который открыт ssh порт
 pdsh -w root@192.168.10.[10-100] "uptime" | sort -n   # параллельная работа с несколькими машинами
```

5.12 QoS

Ограничение скорости отдачи (Limit upload)

Целесообразно использовать для взаимодействия с DSL устройствами и различными модемами для согласования скорости.

```
# Для модема с upload 512Kbit. 90% от 512Kbit примерно 480Kbit
# tc qdisc add dev eth0 root tbf rate 480kbit latency 50ms burst 1540
```

```
# tc -s qdisc ls dev eth0                         # Статус
# tc qdisc del dev eth0 root                      # Удалить очередь
# tc qdisc change dev eth0 root tbf rate 220kbit latency 50ms burst 1540
```

Качество обслуживания (QoS)

tc - используем для приортета VoIP. Пусть VoIP спользует udp на портах 10000:11024 и интерфейс eth0. Команды в примере зададут качество обслуживания для 3 очередей и выделят VoIP для очереди 1 с QoS 0x1e (установлены все биты). По умолчанию потоков в очереди 3 и QoS Minimize-Delay поток в очереди 2.

```
# tc qdisc add dev eth0 root handle 1: prio priomap 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 0
# tc qdisc add dev eth0 parent 1:1 handle 10: sfq
# tc qdisc add dev eth0 parent 1:2 handle 20: sfq
# tc qdisc add dev eth0 parent 1:3 handle 30: sfq
# tc filter add dev eth0 protocol ip parent 1: prio 1 u32 \
  match ip dport 10000 0x3C00 flowid 1:1          # используем диапазон портов сервера
  match ip dst 195.96.0.1 flowid 1:1              # и/или сервер IP
```

Статус и удаление QoS

```
# tc -s qdisc ls dev eth0                         # Состояние очереди
# tc qdisc del dev eth0 root                      # Удалить все QoS
```

Расчет диапазона портов и маски: tc фильтр определяет диапазон портов с портом и маской, которые мы будемем использовать для расчета. Определим 2^N границы диапазона портов, расчитаем диапазон и переведем в HEX. Это и есть маска. Например для 10000 -> 11024, диапазон 1024.

```
# 2^13 (8192) < 10000 < 2^14 (16384)              # граница 2^14 = 16384
# echo "obase=16;(2^14)-1024" | bc                # маска 0x3C00
```

5.13 NIS

```
# ypwhich                  # вернуть имя сервера YP с оригиналом базы данных
# domainname               # NIS domain name
# ypcat group              # Показать группы с NIS сервера
# cd /var/yp && make       # Перестроить YP базу
# rpcinfo -p servername    # Отчет о RPC сервисах на сервере

# ps auxww | grep ypbind   # статус NIS
/usr/sbin/ypbind
# yppoll passwd.byname
Map passwd.byname has order number 1190635041. Mon Sep 24 13:57:21 2007
The master server is servername.domain.net.

# cat /etc/yp.conf
ypserver servername
domain domain.net broadcast
```

5.14 Netcat (nc)

Создаёт или читает/пишет TCP/IP соединения

Передать файл

Копирование файлов по сети напрямую по tcp. Копирование быстрое и не нужно создвать ftp, smb и т.д, мы просто делаем файлы доступными по сети. Пусть 192.168.1.1 IP сервера.

```
# Передача папки
server# tar -cf - -C VIDEO_TS . | nc -l -p 4444         # Сделаем архив папки и прикрепим на 4444 порт
client# nc 192.168.1.1 4444 | tar xpf - -C VIDEO_TS     # Получим файл с порта 4444 и распакуем
server# tar -czf - /etc/ | nc -l 3333                   # Быстрый backup
client# nc 192.168.1.1 3333 | pv -b > mybackup.tar.gz
# Передача файла
server# cat largefile | nc -l 5678                      # Публикуем файл largefile на 5678 порт
client# nc 192.168.1.1 5678 > largefile                 # Принимаем файл с 5678 порта в новый файл largefile
server#cat backup.iso | pv -b | nc -l 3333              # Передаем файл с информацией о состоянии (прогресс)
client#nc 192.168.1.1 3333 | pv -b > backup.iso         # Получаем файл с информацией о состоянии (прогресс)
# Передача образа
server# dd if=/dev/sda0 | nc -l 4444                    # Передаем образ раздела
client# nc 192.168.1.1 4444 | dd of=/dev/sda0           # Клонируем раздел
client# nc 192.168.1.1 4444 | dd of=sda0.img            # или записываем образ в файл
server# dd if=/dev/hdb5 | gzip -9 | nc -l 3333
client# nc 192.168.1.1 3333 | pv -b > myhdb5partition.img.gz
```

Сканер портов

```
# nc -v -w 1 localhost -z 1-5901 |grep succeeded!
Connection to localhost 22 port [tcp/ssh] succeeded!
Connection to localhost 25 port [tcp/smtp] succeeded!
# nc -z 192.168.1.112 1-90
Connection to 192.168.1.112 22 port [tcp/ssh] succeeded!
Connection to 192.168.1.112 80 port [tcp/http] succeeded!
```

Чат

```
zabrosov# nc -lp 4444
boss    # nc 192.168.1.11 4444
```

Hacks

```
# nc -lp 4444 -e /bin/bash                        # Удаленый shell (server backdoor)
# nc -lp 4444 -e cmd.exe                          # Удаленый shell для Windows; -e только для win ?
# webserver
# while true; do nc -l -p 80 < zabrosov.html; done # Петля на 80 порту, отдаёт html страничку
```

```
# Бэкап по шифрованному тунелю, с автозакрытием в конце операции
server# cat backup.iso | nc -l 3333
client# ssh -f -L 23333:127.0.0.1:3333 zabrosov@192.168.1.1 sleep 10; nc 127.0.0.1 23333 | pv -b > backup.iso
```

6 SSH SCP

6.1 Public key | 6.2 Проверка подписи | 6.3 SCP | 6.4 Туннелирование | 6.5 SSH tricks

6.1 Public key аутентификация

Аутентификация - подтверждение подлинности; установление соответствия лица названному им идентификатору.
Авторизация - процесс предоставления определенному лицу прав на выполнение некоторых действий.
Идентификация - присвоение субъектам и объектам идентификатора и/или сравнение идентификатора с перечнем присвоенных идентификаторов.

Используем *ssh-keygen* для создания пары ключей. ~/.ssh/id_dsa приватный ключ, ~/.ssh/id_dsa.pub публичный ключ.
Копируем публичный ключ на сервер и добовляем его в файл ~/.ssh/authorized_keys2 в вашу домашнюю папку.

```
# ssh-keygen -t dsa -N ''
# cat ~/.ssh/id_dsa.pub | ssh you@host-server "cat - >> ~/.ssh/authorized_keys2"
```

Соединяемся с хостом без пароля используя public key аутентификацию

```
# ssh-keygen -q                                          # Генерируем пару ключей
# nmap -sX -p 22 192.168.10.10-250 -oN /data/host.txt    # Сканирование машин с ip от 10 по 250 \
                                                         # на порт ssh с записью в файл
# grep  192.168.10 /data/host.txt > /data/host           # Пишем строки с ip адресами в файл
# sed 's/Interesting ports on //g' /data/host | sed 's/://g' |sed 'w /data/host.txt'
# Копируем публичный ключ на каждую машиу из списка /data/host.txt
# cat /data/host.txt | while read i ; do ssh-copy-id -i ~/.ssh/id_rsa.pub root@"$i" ; done
# Копируем /etc/hosts сразу на все хосты из списка
# cat /data/host.txt | while read i ; do scp /etc/hosts root@"$i":/etc/ ; done
```

6.2 Проверка подписи (fingerprint)

При первом логине на хост, ssh сппросит сохранить ли подпись хоста и является ли этот хост доверенным. Если мы сомневаемся и подозреваем хост в man-in-the-middle attack, то администратор хоста, может отправить вам подпись для сравнения. Получить ей можно с помощью команды *ssh-keygen -l*

```
host# ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub      # RSA key
2048 47:a5:c6:27:78:06:89:f8:97:3d:02:90:17:29:96:a5 /etc/ssh/ssh_host_rsa_key.pub
host# ssh-keygen -l -f /etc/ssh/ssh_host_dsa_key.pub      # DSA key (default)
1024 46:b8:07:38:7a:23:ba:bd:d5:b4:ba:27:cd:a9:38:e5 /etc/ssh/ssh_host_dsa_key.pub

ssh host
The authenticity of host '192.168.50.55 (192.168.50.55)' can't be established.
RSA key fingerprint is 47:a5:c6:27:78:06:89:f8:97:3d:02:90:17:29:96:a5.
Are you sure you want to continue connecting (yes/no)?
```

6.3 Безопасная передача файлов

```
# scp file.txt host-two:/tmp
# scp joe@host-two:/www/*.html /www/tmp
# scp -r joe@host-two:/www /www/tmp        # Копируем папки, -r рекурсия
# с помощью опции -P, указываем на какой порт подключаемся, если ssh на 2525 порту.
# scp -P 2525 ./file_to_copy root@remote_host:/tmp/copied_file
```

6.4 Туннелирование (Tunneling)

SSH туннелирование позволяет перенаправлять порты через SSH соединения, тем самым обеспечивая трафик и доступ к портам, которые иначе были бы заблокированы. Работает только для TCP.

```
# localhost:localport <- destport:gate
# ssh -L localport:desthost:destport user@gate  # Хост назначения будет представлять из себя локальный порт
# localhost:localport -> destport:gate
# ssh -R destport:desthost:localport user@gate  # Локальный порт будет перенаправлен на указанный порт удаленного хоста
# ssh -X user@gate                              # Форвардинг X сеанса
```

Прямое перенаправление на шлюз

Нужно получить доступ к CVS (порт 2401) и http (порт 80), запущенных на удаленном хосте. Мы подключаем к локальному порту 2401, соответствующий порт удаленного хоста, а для доступа к удаленному порту 80 используем локальный порт 8080. Единожды открыв ssh сессию, все соответствующие сервисы удаленного хоста, будут доступны на локальных портах.

```
# ssh -L 2401:localhost:2401 -L 8080:localhost:80 user@gate
```

Форвардинг портов Netbios и RDP

```
# ssh -L 139:smbserver:139 -L 3388:smbserver:3389 user@gate
```

Теперь SMB шара доступна по адресу \\127.0.0.1\, но только если отключена локальные шара, поскольку локальная шара тоже использует 139 порт. Что-бы иметь возможность

оставить локальные расшаренные русурсы включенными, нужно создать новый виртуальный интерфейс с новым IP адресом для создания туннеля, SMB шара будет подключена через этот IP адрес. Кроме того, локальный RDP уже использует порт 3389, поэтому мы выбираем 3388. Для приведенного ниже примера, будем использовать виртуальный IP адрес 10.0.0.1.
Создаем loopback интерфейс с IP адресом 10.1.1.1:

```
# System->Control Panel->Add Hardware # Добавляем новое устройство
# Устанавливаем устройство выбрав его вручную Network adapters -> Microsoft -> Microsoft Loopback Adapter.
Настраиваем IP адрес созданного устройства на 10.1.1.1 маска 255.255.255.0, без шлюза.
advanced-> WINS, Enable LMHosts Lookup; Disable NetBIOS over TCP/IP.
# Enable Client for Microsoft Networks. # Отключить общие файлы и принтеры для сетей Microsoft
```

Подключение клиентов, находящихся за NAT

Есть две машины, находящиеся за NAT шлюзом, клиенты имеют доступ к Linux-шлюзу по ssh. Поскольку будут использованы порты, выше 1024, root доступ не понадобится. На шлюзе мы используем порт 2022.

```
 cliuser# ssh -R 2022:localhost:22 user@gate            # Форвардинг порта клиента 22, на порт 2022, шлюза
cliadmin# ssh -L 3022:localhost:2022 admin@gate         # Форвардинг порта клиента 3022, на порт 2022 шлюза
cliadmin# ssh -p 3022 admin@localhost                   # local:3022 -> gate:2022 -> client:22
```

Подключение к рабочему столу, расположенному за NAT

Нужно получить доступ к Windows клиенту с VNC слушающем на 5900 порту.

```
  cliwin# ssh -R 15900:localhost:5900 user@gate
cliadmin# ssh -L 5900:localhost:15900 admin@gate
cliadmin# vncconnect -display :0 localhost
```

Multi-hop ssh tunnel

Мы не можете получить прямой доступ к ssh сервера,а возможно это только через промежуточные хосты(например из-за проблем с маршрутизацией), но получить соединение клиент-сервер необходимо, к примеру что-бы скопировать файлы через SCP или пробросить порт для SMB. Сделать это можно, организовав туннель из цепочки хостов. Допустим нам нужно перебросить ssh порт клиента к серверу, в два скачка. Когда туннель будет создан, будет возможно прямое подключение клиент - сервер.
Создание туннеля: cli -> host_1 -> host_2 -> сервер and dig tunnel 5678

```
   cli# ssh -L 5678:localhost:5678 host_1        # 5678 произвольный порт для туннеля
host_1# ssh -L 5678:localhost:5678 host_2        # 5678 с host_1 на host_2
host_2# ssh -L 5678:localhost:22 server          # и туннель на порт 22 сервера
```

client -> server используя туннель и порт 5678

```
# ssh -p 5678 localhost                          # Соединение напрямую cli -> server
# scp -P 5678 myfile localhost:/tmp/             # Копируем файлы через туннель
# rsync -e 'ssh -p 5678' myfile localhost:/tmp/ # или rsync
```

Автосоединение (Autoconnect)

/home/admin/port_forward.sh

```
#!/bin/sh
COMMAND="ssh -N -f -g -R 3022:localhost:22 admin@gate"
pgrep -f -x "$COMMAND" > /dev/null 2>&1 || $COMMAND
exit 0
```

crontab -e

```
1 * * * * admin /home/admin/port_forward.sh # Поддерживаем соединение

# Другой вариант, не проверял
# while [ ! -f /tmp/stop ];
do ssh -o ExitOnForwardFailure=yes -R 2222:localhost:22 target "while nc -zv localhost 2222;
do sleep 5; done"; sleep 5;done
```

6.5 SSH tricks

```
# ssh-copy-id user@host         # Копируем ключ на host, для доступа на host без пароля
# cat ~/.ssh/id_rsa.pub | ssh user@machine "mkdir ~/.ssh; cat >> ~/.ssh/authorized_keys" # Копируем ключ без ssh-copy-id
# ssh -4 -C -c blowfish-cbc     # Быстрый и безопасный ssh клиент
# ssh-keygen; ssh-copy-id user@host; ssh user@host # Передаем public key одной командой
# sed -i 8d ~/.ssh/known_hosts  # Удалить строку 8, если "ssh host key change"

# ssh -N -L2001:localhost:80 user@host # Туннель с host:80 к вашей машине , порт 2001. http://localhost:2001/
# dd if=/dev/dsp |ssh -c arcfour -C username@host dd of=/dev/dsp # Передача звука микрофона на удаленный host
# ssh user1@local_server 'play /usr/share/sounds/gaim/arrive.wav # Удаленное выполнение команды
# ssh user@host cat /path/to/remotefile |diff /path/to/localfile # Сравнение локального и удаленного файлов
# sshfs name@server:/path/to/folder /path/to/mount/point         # Монтируем через ssh
# ssh -t reachable_host ssh unreachable_host                     # Соеденение через хост
# Копируем с host1 на host2 через наш компьютер
# ssh root@host1 "cd /somedir/tocopy/ && tar -cf - ." | ssh root@host2 "cd /samedir/tocopyto/ && tar -xf -"
# ssh -fX user@host firefox        # Удаленный запуск GUI программ
# ssh -t remote_host screen -r     # screen через ssh
# ssh host -l user "`cat cmd.txt`" # Комплексное выполнение команд через ssh

# mysqldump -add-drop-table –extended-insert –force –log-error=error.log -uUSER -pPASS OLD_DB_NAME
 | ssh -C user@newhost "mysql -uUSER -pPASS NEW_DB_NAME" # Dumps MySQL DB на новый сервер

# yes | pv | ssh $host "cat > /dev/null"                      # Скорость ssh соединения, yum install pv
# ssh -t user@some.domain.com /usr/bin/screen -xRR            #Сохранить удаленную сессию screen для re-connect
# rsync –partial –progress –rsh=ssh $file_source $user@$host:$destination_file # Копируем большой файл. rsync на обоих машинах
# autossh -M50000 -t server.example.com 'screen -raAd mysession'  # SSH сессия открыта всегда, помогает если теряется соединение
```

```
# tar -cj /backup |cstream -t 777k |ssh host 'tar -xj -C /backup' # Ограничиваем скорость с cstream до 777k bit/s
# ssh user@host cat /path/to/some/file | xclip                    # Копируем stdin в наш X11 buffer
# while read server; do ssh -n user@$server "command"; done < servers.txt # запускаем команды из списка на сервере
```

Постоянное соединение:

```
# ssh -MNf user@host    # Создать постоянное соединение с машиной, удобно для rsync/sftp/cvs/svn, работает в bg
# Редактируем ~/.ssh/config или /etc/sshd_config
ControlPath ~/.ssh/master-%r@%h:%p
ControlMaster no
```

Port Knocking

```
# knock host 3000 4000 5000 && ssh -p port user@host && knock host 5000 4000 3000

# Установить knock. Обращаемся к определенным портам, чтоб от крыть доступ к ssh, а потом его закрыть.
# nano /etc/knockd.conf:
[options]
logfile = /var/log/knockd.log
[openSSH]
sequence = 3000,4000,5000
seq_timeout = 5
command = /sbin/iptables -A INPUT -i eth0 -s %IP% -p tcp -dport 22 -j ACCEPT
tcpflags = syn
[closeSSH]
sequence = 5000,4000,3000
seq_timeout = 5
command = /sbin/iptables -D INPUT -i eth0 -s %IP% -p tcp -dport 22 -j ACCEPT
tcpflags = syn
```

7 VPN с SSH

7.1 P-2-P | 7.2 Lan-2-Lan

OpenSSH поддерживает устройства tun/tap, позволяющие создавать шифрованный туннель. Плюс протокола SSH в том, что для реализации не нужно устанавливать и настраивать дополнительный софт, минус - низкая производительность на медленных линиях.
В /etc/ssh/sshd_config, должны стоять опции:

```
PermitRootLogin yes
PermitTunnel yes
```

7.1 P-2-P

Соединим два хоста client и server. Соединение инициирует client к server, при этом он должен обладать правами root. Конечные адреса туннеля 10.0.0.1 (server) и 10.0.0.2 (client), кроме того мы создаем устройство tun1.

```
# modprobe tun                                          # Проверка поддержки tun ядром

client# ssh -w1:1 root@server
server# ifconfig tun1 10.0.1.1 netmask 255.255.255.252   # Выполняем в server shell

client# ifconfig tun1 10.0.1.2 netmask 255.255.255.252   # SSH клиент
```

7.2 Lan-2-Lan

Есть две сети, их нужно соединить, сеть А с адресом 192.168.51.0/24 и сеть Б с адресом 192.168.16.0/24. 192.168.51.0/24 (сеть А)|шлюз А <-> шлюз Б|192.168.16.0/24 (сеть Б)

```
# сеть B
gateA># ssh -w5:5 root@gateB
gateB># ifconfig tun5 10.0.1.1 netmask 255.255.255.252
gateB># route add -net 192.168.51.0 netmask 255.255.255.0 dev tun5
gateB># echo 1 > /proc/sys/net/ipv4/ip_forward
gateB># iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# сеть A
gateA># ifconfig tun5 10.0.1.2 netmask 255.255.255.252
gateA># route add -net 192.168.16.0 netmask 255.255.255.0 dev tun5
gateA># echo 1 > /proc/sys/net/ipv4/ip_forward
gateA># iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

8 Backup

8.1 rsync | 8.2 tar | 8.3 dd

8.1 rsync

rsync используется для удаленного копирования (локально тоже используется) или синхронизации файлов и каталогов. Может практически целиком заменить cp и scp, поддерживает сжатие и рекурсию, прерванные передачи можно перезапустить.

```
# rsync -a /home/zabrosov/ /backup/zabrosov/          # "archive" mode
# rsync -a /var/ /var_bak/
# rsync -aR --delete-during /home/user/ /backup/      # Используется относительный путь
```

Копируем через сет с компрессией. По-умолчанию, Rsync использует для передачи протокол SSH в том числе и с ключами, если таковые имеются. Символ ":" используется как в SCP.

```
rsync -axSRzv /home/user/ user@server:/backup/user/
```

Исключить из процесса удаленного копирования директорию tmp в /home/user/ и сохранить иерархию, удаленная директория будет иметь структуру /backup/home/user/:

```
# rsync -azR --exclude /tmp/ /home/user/ user@server:/backup/
# rsync -az -e 'ssh -p 20022' /home/colin/ user@server:/backup/colin/ # Использовать 20022 порт для SSH
```

Можно использовать *rsync* с "::", это гораздо быстрее, но трафик не шифруется. Местонахождение папки для резервного копирования (например /backup) можно настроить в файле /etc/rsyncd.conf. Переменная RSYNC_PASSWORD служит для того, что-бы избежать необходимости ввода пароля вручную.

```
# rsync -axSRz /home/ user@hostname::rmodule/backup/
# rsync -axSRz user@hostname::rmodule/backup/ /home/    # Обратно

   # Опции:
   -a, --archive         # режим архивирования; как -rlptgoD (без -H)
   -r, --recursive       # обходить директории (рекурсия)
   -R, --relative        # относительные пути
   -H, --hard-links      # сохранять жесткие ссылки ( hardlink )
   -S, --sparse          # эффективная обработка файлов
   -x, --one-file-system # не пересекать границы файловой системы
   --exclude=PATTERN     # исключить файлы заданного образца
   --delete-during       # удаление файлов при передаче (источник)
   --delete-after        # удаление файлов после передачи
```

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 # Создалbv 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/
```

8.3 dd

dd (disk dump или destroy disk) - используется для копирования (конвертирования) дисков, разделов, и прочих операций копирования.
Синтаксис:

```
# dd if=<source> of=<target> bs=<byte size> conv=<conversion>
# Опции conv:
  notrunc # Не обрезать нули в файле на выходе, записывая их как нули
  noerror # Продолжать после ошибок чтения
  sync    # Дополнять каждый входящий блок нулями до размера ibs-size
```

Размер входных данных по-умолчанию 512 байт (1 блок). Увеличение размера блока ускоряет процесс копирования, но требует больше памяти.

Резервное копирование и восстановление

```
# dd if=/dev/hda of=/dev/hdc bs=16065b               # Копировать с диска на диск с таким-же размером
# dd if=/dev/sda7 of=/home/root.img bs=4096 conv=notrunc,noerror # Резервное копирование в файл образа
# dd if=/home/root.img of=/dev/sda7 bs=4096 conv=notrunc,noerror # Восстановление из файла образа
# dd bs=1M if=/dev/ad4s3e | gzip -c > ad4s3e.gz                 # Сделать резервную копию и заархивировать в Zip
# gunzip -dc ad4s3e.gz | dd of=/dev/ad0s3e bs=1M                # Восстановить из архива
# dd bs=1M if=/dev/ad4s3e | gzip | ssh eedcoba@fry 'dd of=ad4s3e.gz' # Что и выше, удаленно
# gunzip -dc ad4s3e.gz | ssh root@host 'dd of=/dev/ad0s3e bs=1M'
# dd if=/dev/ad0 of=/dev/ad2 skip=1 seek=1 bs=4k conv=noerror    # Пропустить MBR (Master Boot Record)
                                                                # Необходимо если диск назначения (ad2) меньше
```

Recover

dd считывает раздел поблочно, если на диске предположительно есть проблемы, нужно использовать опцию *conv=sync,noerror*, при этом dd будет пропускать битые блоки и записывать нули на диск назначения. Поэтому важно, установить размер блока, равным, или меньшим, чем размер блока на диске. Вполне подходящим будет размер блока в 1 килобайт, установить размер на входе и выходе можно опцией *bs=1k*. Если на диске имеются сбойные сектора, но основные данные нужно сохранить с данного раздела, можно

создать файл образа, смонтировать образ и копировать данные на новый диск. С установленной опцией *noerror*, dd пропустит поврежденные блоки, записав на их место нули, при этом, потеряны будут, только данные, содержавшиеся в сбойных секторах диска.

```
# dd if=/dev/hda of=/dev/null bs=1m                  # Проверить на наличие бэд блоков
# dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc | gzip | ssh root@fry 'dd of=hda1.gz bs=1k' # Отправить на удаленный хост
# dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc of=hda1.img    # Сохранить в образ
# mount -o loop /hda1.img /mnt                       # Создание и монтирование образа
# rsync -ax /mnt/ /newdisk/                          # Копировать на новый диск
# dd if=/dev/hda of=/dev/hda                         # Обновить
# Обновление диска, безопасная операция, но диск при этом должен быть размонтирован
```

Удаление данных

```
# dd if=/dev/zero of=/dev/hdc    # Удалить все данные с диска (забивает диск нулями)
# dd if=/dev/urandom of=/dev/hdc # Удалить все данные с диска (более предпочтительный вариант, но дольше)
# kill -USR1 PID                 # Посмотреть текущее состояние dd
```

MBR

MBR содержит код загрузчика и таблицу разделов. Первый 466 байт отводятся под загрузчик, 466-512 байт под таблицу размещения разделов.

```
# dd if=/dev/sda of=/mbr_sda.bak bs=512 count=1       # Сделать резервную копию MBR
# dd if=/dev/zero of=/dev/sda bs=512 count=1          # Удалить MBR и таблицу размещения разделов
# dd if=/mbr_sda.bak of=/dev/sda bs=512 count=1       # Восстановить MBR целиком
# dd if=/mbr_sda.bak of=/dev/sda bs=446 count=1       # Восстановить только загрузчик
# dd if=/mbr_sda.bak of=/dev/sda bs=1 count=64 skip=446 seek=446 # Восстановить таблицу размещения разделов
```

9 sudo

sudo - повысить уровень привелегий, дать права на выполнение, без выдачи пароля на root.

```
# sudo /etc/init.d/dhcpd restart            # Запустить стартовый скрипт от имени root
# sudo -u sysadmin whoami                   # Запустить команду от имени другого пользователя
```

sudo конфигурируется в файле /etc/sudoers или с помощью команды *visudo*

```
user hosts = (runas) commands          # В /etc/sudoers, базовый синтаксис. разделитель: ,

  users - один или более пользователей или %group (например %wheel) для расширения прав доступа
  hosts - список хостов (или ALL)
  runas - список пользователей (или ALL) от чьего имени могут выполняться команды. Заключается в ( )
  commands - список команд (или ALL), которые можно запустить от имени root или от имени других пользователей (runas)

Используются псевдонимы: User_Alias, Host_Alias, Runas_Alias и Cmnd_Alias.
```

/etc/sudoers:

```
# Псевдонимы хоста, подсети или имена хостов
Host_Alias   DMZ     = 212.118.81.40/28
Host_Alias   DESKTOP = work1, work2

# Псевдонимы пользователя, список пользователей имеющих некоторые права доступа
User_Alias   ADMINS  = colin, luca, admin
User_Alias   DEVEL   = joe, jack, julia
Runas_Alias  DBA     = oracle,pgsql

# Псевдонимы команд, список полных путей до команд
Cmnd_Alias   SYSTEM  = /sbin/reboot,/usr/bin/kill,/sbin/halt,/sbin/shutdown,/etc/init.d/
Cmnd_Alias   PW      = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root # HE root pwd
Cmnd_Alias   DEBUG   = /usr/sbin/tcpdump,/usr/bin/wireshark,/usr/bin/nmap

# Актуальные права доступа
root,ADMINS  ALL     = (ALL) NOPASSWD: ALL    # ADMINS может что-то делать без пароля
DEVEL        DESKTOP = (ALL) NOPASSWD: ALL    # Разработчики имеют полные права доступа на рабочих станциях
DEVEL        DMZ     = (ALL) NOPASSWD: DEBUG  # Разработчики могут отлаживать DMZ сервера

# Пользователь sysadmin может использовать некоторые команды
sysadmin     DMZ     = (ALL) NOPASSWD: SYSTEM,PW,DEBUG
sysadmin     ALL,!DMZ = (ALL) NOPASSWD: ALL   # Какие-то права за рамками DMZ
%dba         ALL     = (DBA) ALL              # Группа dba может работать от имени пользователя базы данных

# Все могут монтировать/размонтировать CDROM на рабочих станциях
ALL          DESKTOP = NOPASSWD: /sbin/mount /cdrom,/sbin/umount /cdrom
```

10 Управление пакетами

10.1 yum rpm apt-get | 10.2 repo | 10.3 update kernel

10.1 yum rpm apt-get

yum устанавливает пакет и все зависимости (пакеты|библиотеки) для его работы из репозиториев.

```
# yum install firefox                 # Установит браузер
# yum localinstall openssh-server.i686.5.5p1-24.fc14.2.rpm --nogpgcheck # Установить из локального rpm файла,
                                                                        # разрешить зависимости, не проверять подпись
# yum reinstall openssh-server        # Переустановить пакет
# yum remove openssh-server           # Удалить пакет и, если нужно, его зависимости
# yum update bind                     # Обновление пакета

# yum search ssh      # Поиск по описаниям, именам пакета и другой доступной информации в репозиториях и в локальной базе
```

```
# yum list *firefox*  # Поиск пакетов содержащих в имени firefox в локальной базе и репозиториях
# yum list installed *firefox*  # Поиск среди установленных в системе пакетов содержащих в имени firefox
# yum info *firefox* # Поиск пакетов содержащих в имени firefox (локальной базе и репозиториях) с выводом информации о пакете
# yum whatprovides */bin/ls     # Поиск пакетов необходимых ls

yum clean dbcache|all # Информация кэшируется, очистить кэш
```

Работа с пакетами, без разрешения зависимостей:

```
# rpm -q package         # Имя пакета и версия
# rpm -qa package        # Все установленные пакеты содержащие package
# rpm -K package         # Проверить подписи пакета
# rpm -qf filename       # Пакет содержащий filenames
# rpm -ivh openssh*      # Установить пакет
# rpm -qi bash           # Информация
# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY # Импортировать файл подписи

# Debian|Ubuntu
# apt-cache search nginx             # Поик нужного пакета в репозитарии
# apt-get update                     # Обновить список пакетов
# apt-get install emacs              # Установить пакет emacs
# dpkg --remove emacs                # Удалить пакет emacs
# dpkg -S file                       # Найти какому пакету принадлежит файл
# dpkg -l                            # Список всех установленных пакетов


# ldd /usr/bin/rsync                 # Список необходимых библиотек для rsync
# ldconfig -n /path/to/libs/         # Добавить путь к разделяемым библиотекам
# LD_LIBRARY_PATH                    # Данная переменная устанавливает путь к библиотекам
```

10.2 репозитории

Создаем файл заканчивающийся на *.repo* в */etc/yum.repos.d/* (Для дистрибутивов с yum):

```
# nano /etc/yum.repos.d/google.repo
[Google]                                          # repo-name
name=Google - i386                                # Описание
baseurl=http://dl.google.com/linux/rpm/stable/i386 # протокол и путь к репозиторию\директории
enabled=1                                         # 1-разрешить|0-запретить использование репозитория
gpgcheck=0                                        # 1-разрешить|0-запретить проверку подписи
```

10.3 update kernel, обновить OS

Обновить дистрибутив

Обновим Centos 5.5 до 5.6:

```
# yum clean all
# yum update glibc\*
# yum update yum\* rpm\* pyth\*
# yum clean all
# yum update mkinitrd nash
# yum update selinux\*
# yum update
# shutdown -r now
```

Обновить ядро

Установим ядро с поддержкой виртуализации:

```
# nano /etc/sysconfig/selinux && reboot  # Отключим selinux и перегрузим
  SELINUX=disabled
# yum install kernel-xen xen             # Установка ядра с поддержкой виртуализации и Xen

# nano /boot/grub/menu.lst               # Редактируем меню выбора загрузки версии ядра
  default=0                              # По умолчанию идет загрузка самого первого ядра (xen)
  timeout=5                              # Время ожидания grub
  splashimage=(hd0,0)/grub/splash.xpm.gz
  hiddenmenu
title CentOS (2.6.18-164.6.1.el5xen)
```

Обновить ядро:

```
# yum update kernel-PAE
# yum list installed kernel\*
```

11 Полезные команды

11.1 mail | 11.2 screen | 11.3 find | 11.4 sed | 11.5 awk | 11.6 regexp | 11.7 Доп. команды

11.1 mail

mail - это базовое приложение для отправки и получения электронной почты. Для отправки почты, просто наберите "mail user@domain". Первая строка, это "Тема" (subject), далее идет содержимое письма. Закончить набор и отправить письмо можно, введя на новой строке символ "."(точка):

```
# mail mail@zabrosov.ru
Subject: Servers:We are working!
"All servers is work, well."
.
EOT
#
```

Другой вариант:

```
# echo "This is the mail body" | mail mail@zabrosov.ru
```

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 сисntаксис
# 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 '{}'
```

11.4 sed

sed - это неинтерактивный строчный редактор. (см. regexp)

```
# sed 's/string1/string2/g'                    # Заменить string1 на string2
# cat ./wrong.txt | sed 's/wrong/right/g' > ./right.txt # Вывести содержимое файла, заменить слова и записать в другой файл
# sed 's/\(.*\)1/\12/g'                        # Модифицировать "строку1" в "строку2"
# sed '//,/<\/p>/d' t.xhtml                    # Удалить строки, начинающиеся с p>
                                               # И заканчивающиеся </p>
# sed '/ *#/d; /^ *$/d'                        # Удалить комментарии и пустые строки
# sed 's/[ \t]*$//'                            # Удалить символы табуляции
# sed 's/^[ \t]*//;s/[ \t]*$//'                # Удалить пробелы в начале и конце
# sed 's/[^*]/[&]/'                            # Заключить первый символ в квадратные скобки
# sed = file | sed 'N;s/\n/\t/'                # Порядковый номер в каждой строке
```

11.5 awk

awk - это весьма мощьный и полезный язык для обработки текстовой информации

```
# awk '{ print $2, $1 }' file                  # Вывести из файла 2 колонки, поменя из местами
# awk '{printf("%5d : %s\n", NR,$0)}' file     # Форматирование вывода с номерами строк
# awk '{print FNR "\t" $0}' files              # Несколько измененный вариант
# awk NF test.txt                              # Удалить из вывода пустые строки (аналогично grep '.')
# awk 'length > 80'                            # Напечатать строки, длинной более 80 символов
```

11.6 Regexp

Regexp - регулярные выражения

```
[\^$.|?*+()           # Специальные символы, остальные символы означают самих себя
\                     # Экранирует специальные символы
*                     # Повтор 0 или 1 раз
.                     # Любой символ, за исключением символа новой строки
.*                    # Совпадает 0 или более символов
^                     # Начало строки
$                     # Конец строки
.$                    # Совпадает с одним любым символ в конце строки
^ $                   # Совпадает со строкой, состоящей из одного пробела
[^A-Z]                # Любые символы, не входящие в диапазон от A до Z
```

11.7 Доп. команды

```
# sort -t. -k1,1n -k2,2n -k3,3n -k4,4n         # Отсортировать IPv4 ip адреса
# echo 'Test' | tr '[:lower:]' '[:upper:]'     # Смена регистра символов
# echo foo.bar | cut -d . -f 1                 # Вернет foo
# PID=$(ps | grep script.sh | grep bin | awk '{print $1}')    # PID запущенного скрипта
# PID=$(ps axww | grep [p]ing | awk '{print $1}')             # PID процесса ping
# IP=$(ifconfig $INTERFACE | sed '/.*inet addr:/!d;s///;s/ .*//')
# cat /etc/master.passwd | grep -v root | grep -v \*: | awk -F":" \ # Создание файла паролей http passwd
# '{ printf("%s:%s\n", $1, $2) }' > /usr/local/etc/apache2/passwd

# testuser=$(cat /usr/local/etc/apache2/passwd | grep -v \    # Проверить пользователя в passwd
# root | grep -v \*: | awk -F":" '{ printf("%s\n", $1) }' | grep ^user$)
# :(){ :|:& };:                                # bash fork bomb :). Машина зависнет!
# tail +2 file > file2                         # Удалить первую строку из файла
```

Изменить расширение для кучи файлов

```
# ls *.cxx | awk -F. '{print "mv "$0" "$1".cpp"}' | sh # .cxx в .cpp
# ls *.c | sed "s/.*/cp & &.$(date "+%Y%m%d")/" | sh   # Копировать файлы *.c в *.c.20080401
# rename .cxx .cpp *.cxx                               # Переименовать все файлы .cxx в cpp
# for i in *.cxx; do mv $i ${i%%.cxx}.cpp; done        # Встроенными средствами
```

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.

13 Печать

```
# lpr exemple.ps                      # Печать на принтер по-умолчанию
# export PRINTER=hp4600               # Сменить принтер по-умолчанию
# lpr -Php4500 #2 page.ps             # Печать 2-х экземпляров, используя принтер hp4500
# lpr -o Duplex=DuplexNoTumble        # Печать двухсторонних страниц
# lpr -o PageSize=A4,Duplex=DuplexNoTumble

# lpq                                 # Проверить очередь печати принтера по-умолчанию
# lpq -l -Php4500                     # Очередь печати принтера hp4500 с отладочной информацией
# lprm -                              # Удалить все пользовательские задания на печать, с принтера по-умолчанию
# lprm -Php4500 3186                  # Удалить задание из очереди печати с номеров 3186
# lpc status                          # Список всех доступных принтеров
# lpc status hp4500                   # Проверка доступности принтера и длины очереди печати
```

14 Дисковые квоты

Квоты распределяются на уровне файловой системы и поддерживаются ядром.
Пакет *quota*, нужно установит дополнительно, затем активировать дисковые квоты в файле fstab и перемонтировать раздел

```
/dev/sda2     /home     ext3     rw,acl,user_xattr,usrquota 1 1
# mount -o remount /home
# mount                               # Проверьте, активна-ли usrquota, иначе перезагрузитесь
```

Инициализация файла *quota.user* с помощью *quotacheck*.

```
# quotacheck -vum /home
# chmod 644 /home/aquota.user         # Позволить пользователю просматривать свои квоты
# quota -v
```

По-умолчанию, дисковые квоты не накладывают никаких ограничений (установлены в 0). Установить необходимые лимиты для пользовательских квот можно с помощью программы edquota. Так-же лимиты можно дублировать на других пользователей. Размер блока по-умочанию, 1 kб. Время действия можно установить с помощью edquota -t

```
# edquota -u colin
Disk quotas for user colin (uid 1007):
  Filesystem         blocks       soft       hard     inodes     soft     hard
  /dev/sda8             108       1000       2000          1        0        0
```

Команда *edquota -p* используется для дублирования квот на других пользователей.

```
# edquota -p refuser `awk -F: '$3 > 499 {print $1}' /etc/passwd`
# edquota -p refuser user1 user2      # Дублируем на 2 пользователей

# quota -u colin                      # Проверить квоты пользователя
# repquota /home                      # Полный отчет по разделу для всех пользователей
```

15 Поиск неисправностей

15.1 Поиск неисправностей | 15.2 Поиск неисправностей

15.1 Поиск неисправностей

```
info
info
```

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
```

17 Шифрование файлов

17.1 OpenSSL | 17.2 GPG

17.1 OpenSSL

Зашифровать и расшифровывать:

```
# openssl aes-128-cbc -salt -in file -out file.aes    # Шифровать файл
# openssl aes-128-cbc -d -salt -in file.aes -out file   # Расшифровать файл

# tar -cf - directory | openssl aes-256-cbc -salt -out directory.tar.aes      # Архивировать и зашифровать директории
# openssl aes-256-cbc -d -salt -in directory.tar.aes | tar -x -f -            # Расшифровать директории и распаковать архив

# tar -zcf - directory | openssl aes-128-cbc -salt -out directory.tar.gz.aes  # Архивировать и зашифровать директории
# openssl aes-128-cbc -d -salt -in directory.tar.gz.aes | tar -xz -f -        # Расшифровать директории и распаковать архив
```

17.2 GPG

GnuPG известный способ шифрования и подписи электронных писем или других данных, кроме того gpg предоставляет расширенную систему управления ключами. В данных примерах рассматривается только шифрование файлов. Самым простым является симметричный шифр. В этом случае файл шифруется с помощью пароля, соответственно расшифровать его может тот, кто знает этот пароль, никаких ключей не требуется. GPG добавляет расширение "*.gpg" к имени зашифрованного файла

```
# gpg -c file                       # Зашифровать файл по паролю
# gpg file.gpg                      # Расшифровать файл ( -o другой файл)
```

Шифрование с использованием ключей

Приватный ключ и публичный ключ, основа ассиметричной криптографии. О чем нужно помнить:

1.Ваш публичный ключ используется другими для шифрования файлов, которые, как получатель, можете расшифровать только вы (даже не тот, кто его шифровал).
2.Ваш приватный ключ зашифрован по паролю и используется для расшифровки файлов, зашифрованных Вашим публичным ключем. Приваиный ключ должен храниться в безопасном месте. Помните, если приватный ключ или пароль от него будут потеряны, вместе с ними пропадут и зашифрованные файлы.
3.Ключевой файл, может содержать несколько ключей.

Вначале нужно сгенерировать пару ключей. Значения по-умолчанию вполне подойдут, однако вам нужно будет ввести имя, адрес электронной почты и комментарий (не обязательно). Комментарий полезен при создании более одного ключа для данного имени/e-mail. Так-же вам нужно будет задать ключевую фразу (именно фразу а не слово).

```
# gpg --gen-key                     # Это может занять некоторое время

~/.gnupg/pubring.gpg                # Содержит ваш публичный ключ а так-же импортируемые ключи
~/.gnupg/secring.gpg                # Может содержать больше одного ключа

# Опции
  -e Зашифровать данные
  -d Расшифровать данные
  -r ИМЯ зашифровать для получателя ИМЯ (или 'полное имя' или 'email@domain')
  -a Создать "ascii armored" вывод ключа
  -o Вывести в файл
```

Шифрование только для персонального использования

Не требует экспорта/импорта какого либо ключа, они у вас уже есть.

```
# gpg -e -r 'Your Name' file                # Зашифровать с помощью публичного ключа
# gpg -o file -d file.gpg                   # Расшифровать. Используется опция -o, иначе пойдкт в stdout
```

Шифрование и расшифровка с использованием ключей

Нам нужно экспортировать ваш публичный ключ, что-бы им могли пользоваться для расшифровки данных. Мы должны импортировать публичный ключ от Alice, что-бы шифровать файлы для нее. Ключи можно передать в обычном ascii файле. Например Alice экспортирует ключ, вы его импортируете себе, теперь вы можете шифровать для нее файлы и расшифровать их сможет только она.

```
# gpg -a -o alicekey.asc --export 'Alice'     # Alice экспортирует ключ в ascii файл
# gpg --send-keys --keyserver subkeys.pgp.net KEYID   # Alice кладет ключ на сервер
# gpg --import alicekey.asc                   # Вы импортируете ключ себе
# gpg --search-keys --keyserver subkeys.pgp.net 'Alice' # Или забираете его на сервере

# gpg -e -r 'Alice' file                      # Зашифровать файл для Alice
# gpg -d file.gpg -o file                     # Расшифровать файл, зашифрованный Alice для вас
```

Управление ключами

```
# gpg --list-keys                           # Список публичных ключей с KEYIDS
# KEYID следует за '/' например для: pub   1024D/D12B77CE - KEYID это D12B77CE
# gpg --gen-revoke 'Your Name'              # Сгенерировать CRL (certificate revocation list)
# gpg --list-secret-keys                    # Список приватных ключей
# gpg --delete-keys NAME                    # Удалмть публичный ключ с локальной "связки ключей"
# gpg --delete-secret-key NAME              # Удалить приватный ключ с локальной "связки ключей"
# gpg --fingerprint KEYID                   # Показать отпечаток ключа
# gpg --edit-key KEYID                      # Редактировать ключ (например подпись или добавить/удалить email)
```

18 Шифрование разделов

18.1 LUKS | 18.2 dm-crypt

18.1 LUKS

Используем Linux dm-crypt (device-mapper) на ядре 2.6. Шифровать будем раздел /dev/sdc1, это может быть любой раздел, диск, USB или файл, созданный losetup. Здесь мы будем использовать /dev/loop0, смотрите Файловая система. Device mapper использует метку для идентификации раздела, в данном примере sdc1, но это может быть любая другая строка.

LUKS с dm-crypt очень удобен для шифрования разделов диска, он позволяет иметь несколько паролей для одного раздела а так-же с легкостью менять их. Что-бы проверить доступно-ли у вас использование LUKS, наберите: *cryptsetup --help*.

```
# Создать раздел
# dd if=/dev/urandom of=/dev/sdc1          # Опционально. Только для параноиков
# cryptsetup -y luksFormat /dev/sdc1       # Это уничтожит все данные на sdc1
# cryptsetup luksOpen /dev/sdc1 sdc1
# mkfs.ext3 /dev/mapper/sdc1               # Будет создана файловая система ext3
# mount -t ext3 /dev/mapper/sdc1 /mnt
# umount /mnt
# cryptsetup luksClose sdc1                # Отсоединить зашифрованный раздел
# Монировать
# cryptsetup luksOpen /dev/sdc1 sdc1
# mount -t ext3 /dev/mapper/sdc1 /mnt
# Размонтировать
# umount /mnt
# cryptsetup luksClose sdc1
```

18.2 dm-crypt

```
# cryptsetup -y create sdc1 /dev/sdc1      # Или любой другой раздел, типа /dev/loop0
```

```
# dmsetup ls                              # Проверить, покажет: sdc1 (254, 0)
# mkfs.ext3 /dev/mapper/sdc1              # Только если делается впервые!
# mount -t ext3 /dev/mapper/sdc1 /mnt
# umount /mnt/
# cryptsetup remove sdc1                  # Отсоединить зашифрованный раздел
```

Делаем тоже самое, (без создания fs), что-бы переподключить раздел. При вводе некорректного пароля команда mount не будет выполнена. В таком случае просто удалите отображение sdc1 (*cryptsetup remove sdc1*) и создайте поновой.