Unix Toolbox по-русски

Это неофициальная русская версия известной работы Колина Баршеля (Colin Barschel) «Тетради юниксоида (Unix Toolbox)». Информацию об авторских правах читайте ниже.

This is unofficial Russian version of the well-known work by Colin Barschel «Unix Toolbox». Would you mind please reading copyright notices below.

Аннотация/Abstract

Документ является коллекцией Unix/Linux/BSD команд и решений полезных для работников ИТ сферы или для опытных пользователей. Это практический гид с краткими пояснениями, тем не менее читателю необходимо понимать что он делает.

This document is a collection of Unix/Linux/BSD commands and tasks which are useful for IT work or for advanced users. This is a practical guide with concise explanations, however the reader is supposed to know what s/he is doing.

Копирайт/Copyright

Содержание/Contents

Сеть

Linux

```
# ethtool eth0                                  # Показать Ethernet статус
# ethtool -s eth0 speed 100 duplex full   # Принудительная установка режима 100Mbit Full duplex
# ethtool -s eth0 autoneg off             # Отключить автоопределение
# ethtool -p eth1                               # Мигать индикатором сетевой карты — если поддерживается
# ip link show                                  # Список сетевых интерфейсов в Linux (подобна ifconfig)
# ip link set eth0 up                           # Активировать сетевой интерфейс (или отключить). Аналог "ifconfig eth0 up"
# ip addr show                                  # Список всех IP адресов в Linux (аналог ifconfig)
# ip neigh show                                 # Тоже что и arp -a
```

Другие операционный системы

```
# ifconfig fxp0                                 # Проверить поле "media" во FreeBSD
# arp -a                                        # Показать таблицу маршрутизации сети, роутера (или хоста) (все OS)
# ping cb.vu                                    # Пинговать хост
# traceroute cb.vu                              # Печатать путь маршрута до точки назначения
# ifconfig fxp0 media 100baseTX mediaopt full-duplex # 100Mbit full duplex (FreeBSD)
# netstat -s                                    # Общесистемная статистика по всем сетевым протоколам
```

Дополнительные инструменты для отладки сети, которые не всегда установлены по умолчанию, но найти их не трудно:

```
# arping 192.168.16.254     # Пропинговать на уровне ethernet
# tcptraceroute -f 5 cb.vu  # Использует tcp вместо icmp что-бы отслеживать маршрут через фаервол
```

Маршрутизация сети

Печать таблицы маршрутизации

```
# route -n                  # Linux или используйте "ip route"
# netstat -rn               # Linux, BSD и UNIX
# route print               # Windows
```

Добавление и удаление маршрута

FreeBSD

```
# route add 212.117.0.0/16 192.168.1.1
# route delete 212.117.0.0/16
# route add default 192.168.1.1
```

Добавить постоянный маршрут сети в */etc/rc.conf*

```
static_routes="myroute"
route_myroute="-net 212.117.0.0/16 192.168.1.1"
```

Linux

```
# route add -net 192.168.20.0 netmask 255.255.255.0 gw 192.168.16.254
# ip route add 192.168.20.0/24 via 192.168.16.254       # Как и выше с ip маршрутом
# route add -net 192.168.20.0 netmask 255.255.255.0 dev eth0
# route add default gw 192.168.51.254
# ip route add default via 192.168.51.254 dev eth0      # Как и выше с ip маршрутом
# route delete -net 192.168.20.0 netmask 255.255.255.0
```

Solaris

```
# route add -net 192.168.20.0 -netmask 255.255.255.0 192.168.16.254
# route add default 192.168.51.254 1                    # 1 = прыгнуть на следующий
шлюз
# route change default 192.168.50.254 1
```

Постоянные записи устанавливаются в */etc/defaultrouter*.

Windows

```
# Route add 192.168.50.0 mask 255.255.255.0 192.168.51.253
# Route add 0.0.0.0 mask 0.0.0.0 192.168.51.254
```

Используйте "*add -p*" что-бы сделать маршрут постоянным.

Настройка дополнительных IP адресов

Linux

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

FreeBSD

```
# ifconfig fxp0 inet 192.168.50.254/24                     # Основной IP адрес
сетевого интерфейса
# ifconfig fxp0 alias 192.168.51.254 netmask 255.255.255.0 # Добавить второй IP адрес
в виде псевдонима
# ifconfig fxp0 -alias 192.168.51.254                      # Удалить псевдоним для
второго IP адреса
```

Постоянные записи в */etc/rc.conf*

ifconfig_fxp0="inet 192.168.50.254 netmask 255.255.255.0"
ifconfig_fxp0_alias0="192.168.51.254 netmask 255.255.255.0"

Solaris

Проверка настроек с *ifconfig -a*

```
# ifconfig hme0 plumb                                      # Установить сетевую карту
# ifconfig hme0 192.168.50.254 netmask 255.255.255.0 up    # Первый IP адрес
# ifconfig hme0:1 192.168.51.254 netmask 255.255.255.0 up  # Второй IP адрес
```

Смена MAC адреса

Для начала вы должны деактивировать сетевой интерфейс.. и не говорите никому для чего вы

хотите сменить MAC...)

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:01:02:03:04:05      # Linux
# ifconfig fxp0 link 00:01:02:03:04:05          # FreeBSD
# ifconfig hme0 ether 00:01:02:03:04:05         # Solaris
# sudo ifconfig en0 ether 00:01:02:03:04:05     # Mac OS X Tiger
# sudo ifconfig en0 lladdr 00:01:02:03:04:05    # Mac OS X Leopard
```

Под Windows существует масса инструментов для смены MAC адреса, например etherchange или обратитесь у гуглу на тему "Mac Makeup", "smac".

Сетевые порты

Список открытых портов:

```
# netstat -an | grep LISTEN
# lsof -i                                   # Ссписок всех интернет соединений Linux
# socklist                                  # Список открытых портов Linux
# sockstat -4                               # Список приложений слушающих на отррытых
портах
# netstat -anp --udp --tcp | grep LISTEN  # Linux
# netstat -tup                              # Список активных соединений
входящие/исходящие Linux
# netstat -tupl                             # Список слушающих портов Linux
# netstat -ano                              # Windows
```

Фаерволы

Linux

```
# iptables -L -n -v                 # Статус
Открыть iptables firewall
# iptables -P INPUT       ACCEPT     # Установить политику по умолчанию лоя цепочки
INPUT — "открыть все"
# iptables -P FORWARD     ACCEPT # что и строкой выше только для цепочки FORWARD
# iptables -P OUTPUT      ACCEPT  # аналогично для цепочки OUTPUT
# iptables -Z                       # Сбросить счетчики во всех цепочках
# iptables -F                       # Сбросить все цепочки
# iptables -X                       # Удалить все цепочки
```

FreeBSD

```
# ipfw show                     # Статус
# ipfw list 65535                   # Epyfnm тип фаервол, закрытый или открытый
# sysctl net.inet.ip.fw.enable=0    # Отключить штатный фаервол IPFW
# sysctl net.inet.ip.fw.enable=1    # Включить штатный фаервол IPFW
```

Форвардинг маршрутов

Linux

Проверить и если нужно, включить форвардинг маршрутов

```
# cat /proc/sys/net/ipv4/ip_forward  # Проверить включен или нет форвардинг, 0=off,
1=on
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

или добавьте в /etc/sysctl.conf:

net.ipv4.ip_forward = 1

FreeBSD

Проверить состояние форвардинга и если нужно включить:

```
# sysctl net.inet.ip.forwarding         # Проверить включен форвардинг или нет,
0=off, 1=on
# sysctl net.inet.ip.forwarding=1
# sysctl net.inet.ip.fastforwarding=1   # Для выделенного маршрута или фаервола
```

Запись в /etc/rc.conf:

gateway_enable="YES" # Установите YES если данный хост является шлюзом

Solaris

```
# ndd -set /dev/ip ip_forwarding 1      # Включить форвардинг маршрутов 0=off, 1=on
```

Трансляция сетевых адресов NAT

Linux

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE    # Включить NAT
# iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 20022 -j DNAT \
--to 192.168.16.44:22           # Перебросить порт 20022 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     # Переброс портов из диапазона 993-995
# ip route flush cache
# iptables -L -t nat            # Проверить статус таблицы NAT
```

FreeBSD

```
# natd -s -m -u -dynamic -f /etc/natd.conf -n fxp0
```

Или добавьте в /etc/rc.conf:

```
firewall_enable="YES"           # YES — Включить фаервол
firewall_type="open"            # Тип фаервола (см. /etc/rc.firewall)
natd_enable="YES"               # Включить natd (если firewall_enable == YES)
natd_interface="tun0"           # Сетевой интерфейс или IP адрес, который будет
использовать natd
natd_flags="-s -m -u -dynamic -f /etc/natd.conf"
```

Для переброски портов:

```
# cat /etc/natd.conf
same_ports yes
use_sockets yes
unregistered_only
# redirect_port tcp insideIP:2300-2399 3300-3399  # Диапазон портов
redirect_port udp 192.168.51.103:7777 7777
```

DNS

В Unix, DNS записи действительны для всех интерфейсов и хранятся в */etc/resolv.conf*. Зона к которой принадлежит хост, так-же хранится в этом файле. Минимальная конфигурация:

```
nameserver 78.31.70.238
search sleepyowl.net intern.lab
domain sleepyowl.net
```

Проверить доменное имя:

```
# hostname -d                           # Аналог dnsdomainname
```

Windows

В Windows, DNS настраивается для каждого интерфейса. Что-бы посмотреть текущую конфигурацию и сбросить DNS кэш, используйте:

```
# ipconfig /?                           # Помощь по использованию команды
# ipconfig /all                         # Посмотреть всю информацию, включая DNS
```

Очистка DNS кэша

Вы можете очистить DNS кэш, но помните, некоторые приложения используют свой, персональный кэш (например Фаерфокс), и на них обнуление не подействует.

```
# /etc/init.d/nscd restart              # Перезапустить nscd (демон кэширования имен)
Linux/BSD/Solaris
# lookupd -flushcache                   # OS X Tiger
# dscacheutil -flushcache               # OS X Leopard и более новые
# ipconfig /flushdns                    # Windows
```

Пересылка DNS запросов

Dig, утилита для проверки настроек DNS. Например, используем для проверки публичный DNS сервер *213.133.105.2 ns.second-ns.de*. Обратите внимание с какого сервера клиент получит ответ (упрощенный ответ).

```
# dig sleepyowl.net
sleepyowl.net.          600     IN      A       78.31.70.238
;; SERVER: 192.168.51.254#53(192.168.51.254)
```

Маршрутизатор 192.168.51.254, прислал в качестве ответа, запись типа A. Запись определенного типа для запроса и DNS сервер, могут быть указаны с символом @:

```
# dig MX google.com
# dig @127.0.0.1 NS sun.com             # Проверить локальный dns сервер
# dig @204.97.212.10 NS MX heise.de     # Запрос к внешнему dns серверу
# dig AXFR @ns1.xname.org cb.vu         # Получить всю зону (пересылка зоны) с dns
сервера
```

Еще одна полезная утилита, *host*:

```
# host -t MX cb.vu                      # Получить записть типа MX (Mail Exchange)
# host -t NS -T sun.com                 # Получить NS запись через TCP соединение
# host -a sleepyowl.net                 # Получить все
```

Обратные запросы

Узнать имя по IP адресу можно с помощью таких утилит как *dig*, *host* или *nslookup*:

```
# dig -x 78.31.70.238
# host 78.31.70.238
# nslookup 78.31.70.238
```

файл /etc/hosts

Отдельные хосты могут быть настроены в файле */etc/hosts*, вместо запуска *named*, для разрешения имени в адрес. Формат следующий:

```
78.31.70.238   sleepyowl.net   sleepyowl
```

Приоритет между файлом *hosts* и *DNS* запросом, может быть сконфигурирован в */etc/nsswitch.conf* И */etc/host.conf*. Подобный файл присутствует и в Windows, и расположен как правило по адресу *c:\windows\system32\drivers\etc*

Протокол динамической адресации сети — DHCP

Linux

Некоторые дистрибутивы (*SuSE*) используют в качестве клиента *dhcpd*. Интерфейс по-умолчанию *eth0*.

```
# dhcpcd -n eth0           # Обновить (не всегда работает)
# dhcpcd -k eth0           # Освободить и выключить
```

Lease (срок аренды — это время, на которое IP адрес может быть выдан определенному хосту сети) и вся информация сохраняется в:

```
/var/lib/dhcpcd/dhcpcd-eth0.info
```

FreeBSD

FreeBSD (и Debian) использует *dhclient*. Для настройки нужного сетевого интерфейса (например *bge0*):

```
# dhclient bge0
```

Срок аренды и вся информация сохраняется в:

```
/var/db/dhclient.leases.bge0
```

Используйте */etc/dhclient.conf* для добавления опций или изменения существующих:

```
# cat /etc/dhclient.conf
interface "rl0" {
prepend domain-name-servers 127.0.0.1;
default domain-name "sleepyowl.net";
supersede domain-name "sleepyowl.net";
}
```

Windows

Dhcp аренда (lease) может быть обновлена с помощью *ipconfig*:

```
# ipconfig /renew          # Обновить все адаптеры
# ipconfig /renew LAN      # Обновить сетевой адаптер с именем "LAN"
```

```
# ipconfig /release WLAN   # Освободить сетевой адаптер с именем "WLAN"
```

Неплохой идеей будет дать сетевым адаптерам более внятные имена

Анализ трафика

Анализ трафика с помощью tcpdump

```
# tcpdump -nl -i bge0 not port ssh and src \(192.168.16.121 or 192.168.16.54\)
# tcpdump -n -i eth1 net 192.168.16.121           # Выборка входящий/исходящий по
одному IP адресу
# tcpdump -n -i eth1 net 192.168.16.0/24          # Выборка входящий/исходящий по
адресу сети
# tcpdump -l > dump && tail -f dump       # Вывод через буфер
# tcpdump -i rl0 -w traffic.rl0                   # Писать заголовки пакетов в
бинарный файл
# tcpdump -i rl0 -s 0 -w traffic.rl0              # Писать в бинарник полные пакеты
# tcpdump -r traffic.rl0                          # Прочитать из файла (так-же для
ethereal) для дальнейшего анализа
# tcpdump port 80                                 # Классические команды
# tcpdump host google.com
# tcpdump -i eth0 -X port \(110 or 143\)          # Проверить защищенность pop или
imap
# tcpdump -n -i eth0 icmp                         # Выборка icmp (ping) пакетов
# tcpdump -i eth0 -s 0 -A port 80 | grep GET      # -s 0 для полных пакетов, -A для
ASCII
```

Некоторые важные опции:

- -A — Печатать текст из пакетов (без заголовков)
- -X — Печатать пакеты в *hex* и *ASCII*
- -l — Включить буферизацию вывода
- -D — Показать все активные сетевые интерфейсы

В операционных системах Windows для анализа трафика можно воспользоваться *windump* с www.winpcap.org.
Windump -D выведет список интерфейсов.

Сканирование сети с помощью программы nmap

Nmap, это многофункциональный сканер безопасности с возможностью определения установленной операционной системы. Работает во всех Unix дистрибутивах, так-же существует версия под Windows. Если вы не просканируете свои сервера, за вас это сделают доброжелатели.)

```
# nmap cb.vu               # Просканировать все зарезервированные порты хоста
# nmap -sP 192.168.16.0/24 # Выяснить какой IP каким хостом используется в сети 0/24
# nmap -sS -sV -O cb.vu    # Провести stealth SYN сканирование с определением типа и
версии OS
PORT      STATE  SERVICE            VERSION
22/tcp    open   ssh                OpenSSH 3.8.1p1 FreeBSD-20060930 (protocol 2.0)
25/tcp    open   smtp               Sendmail smtpd 8.13.6/8.13.6
80/tcp    open   http               Apache httpd 2.0.59 ((FreeBSD) DAV/2 PHP/4.
[...]
Running: FreeBSD 5.X
Uptime 33.120 days (since Fri Aug 31 11:41:04 2007)
```

Другие полезные инструменты: hping, конструктор/анализатор IP пакетов, *fping* (fping.sourceforge.net), проверка хостов round-robin.

Контроль трафика (QoS)

Traffic control управляет очередностью, порядком, планированием и другими параметрами трафика в сети. Следующие примеры, небольшие практические приемы для Linux и FreeBSD позволяющие оптимизировать использование пропускной способности.

Ограничение загрузок (upload)

Linux

Для 512 Кбитного модема.

```
# 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
```

FreeBSD

FreeBSD использует *dummynet* — шейпер трафика, встроенный в штатный фаервол операционной системы, *IPFW* или подгружаемый как модуль ядра FreeBSD. *Pipes*, так называемые трубы для трафика, ограничивают пропускную способность в [K|M]{bit/s|Byte/s}, 0 означает безлимитный.
Например, ограничим пропускную способность в 500 Кбит.

```
# kldload dummynet                                  # Загрузить модуль если необходимо
# ipfw pipe 1 config bw 500Kbit/s                   # Создать трубу с ограничением
трафика 500Кбит/с
# ipfw add pipe 1 ip from me to any                 # Отклонять излишний трафик
```

QoS Quality of service

Linux

Приоритет очередей в *tc* для оптимизации *VoIP* трафика. Полные примеры можно посмотреть на voip-info.org или www.howtoforge.com. Следующий пример демонстрирует использование QoS для VoIP трафика.

```
# 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          # Использовать диапазон портов
match ip dst 123.23.0.1 flowid 1:1              # или/и использовать IP сервера
```

Проверить состояние или удалить:

```
# tc -s qdisc ls dev eth0                           # Проверить состояние очереди
# tc qdisc del dev eth0 root                        # Удалмть все очереди
```

FreeBSD

Максимальная пропускная способность соединения 500Кбит/с, мы назначаем 3 очереди с приоритетами 100:10:1 для *VoIP:ssh:всего остального*, соответственно.

```
# ipfw pipe 1 config bw 500Kbit/s
# ipfw queue 1 config pipe 1 weight 100
```

```
# ipfw queue 2 config pipe 1 weight 10
# ipfw queue 3 config pipe 1 weight 1
# ipfw add 10 queue 1 proto udp dst-port 10000-11024
# ipfw add 11 queue 1 proto udp dst-ip 123.23.0.1 # или/и используем IP
# ipfw add 20 queue 2 dsp-port ssh
# ipfw add 30 queue 3 from me to any              # Все остальное
```

Состояние и удаление:

```
# ipfw list                                       # Посмотреть состояние
# ipfw pipe list                                  # Состояние Pipes
# ipfw flush                                      # Очистить все правила кроме
дефолтовых
```

NIS (Информационная служба сети)

Некоторые команды для конфигурирования *NIS* клиента:

```
# ypwhich                          # Получитьимя имя подключенного NIS сервера
# domainname                       # Доменное имя NIS
# ypcat group                      # Показать группу из NIS
# cd /var/yp && make       # Пересобрать базу данных yp
# rpcinfo -p servername            # Отчет RPC служб сервера
```

Ypbind запущен ?

```
# ps auxww | grep ypbind
/usr/sbin/ypbind -s -m -S servername1,servername2   # FreeBSD
/usr/sbin/ypbind                                        # Linux
# yppoll passwd.byname
Map passwd.byname has order number 1190635041. Mon Sep 24 13:57:21 2007
The master server is servername.domain.net.
```

Linux

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

Сетевая утилита netcat

Netcat (nc), известна так-же как "network Swiss Army Knife", предназначена для создания, чтения, записи TCP/IP соединений. Вот несколько полезных примеров, в сети их можно найти массу, например: тут или тут. Вместо *netcat*, можно использовать сокращение *nc*. Так-же можете посмотреть socat.

Передача файлов

Копирование большого файла через TCP соединение. Передача происходит очень быстро и не требует поднятия NFS, SMB, FTP и т.д... просто сделайте файл доступным на сервере и заберите его с клиента. В данном случае 192.168.1.1, IP адрес сервера.

```
server# tar -cf - -C VIDEO_TS . | nc -l -p 4444      # Заархивировать директорию и
выставить архив на порт 4444
client# nc 192.168.1.1 4444 | tar xpf - -C VIDEO_TS  # забрать файл с порта 4444 и
разархивировать в дмректорию
```

```
server# cat largefile | nc -l 5678                    # Приготовить к отправке один
файл
client# nc 192.168.1.1 5678 > largefile               # Забрать файл
server# dd if=/dev/da0 | nc -l 4444                   # Приготовить к отправке файл
образа раздела
client# nc 192.168.1.1 4444 | dd of=/dev/da0          # Забрать файл образа для
создания дублеката раздела
client# nc 192.168.1.1 4444 | dd of=da0.img           # или забрать файл образа и
сохранить как файл
server#  nc -l 5555 < ./test.file            # Файл выставляется в сокет nc с адресом
192.168.1.1, порт 5555
client#  nc 192.168.1.1 5555 > ~/test.file   # На другой машине, забираем файл, с
192.168.1.1 порт 5555
```

Другие трюки

Тут будьте осторожней, вы открываете доступ к своей системе.

Удаленный shell

Опция -e только для Windows версии или используйте nc 1.10.

```
# nc -lp 4444 -e /bin/bash                            # Предоставить удаленную оболочку
# nc -lp 4444 -e cmd.exe                              # Удаленная оболочка в Windows
```

Аварийный Веб сервер

Обслуживать один файл на 80 порту в цикле.

```
# while true; do nc -l -p 80 < unixtoolbox.xhtml; done
```

Простой TCP чат

Элис и Боб могут общаться через простой TCP сокет. Текст передается по нажатию Enter.

```
alice # nc -lp 4444
bob # nc 192.168.1.1 4444
```

Система

```
# uname -a                                  # Версия операционной системы и ядра (BSD)
# lsb_release -a                            # Информация о релизе (LSB distribution)
# cat /etc/SuSE-release                     # Версия SuSE
# cat /etc/debian_version                   # Версия Debian
```

Используйте */etc/DISTR-release* где *DISTR= lsb (Ubuntu)*, *redhat*, *gentoo*, *mandrake*, *sun (Solaris)* и т.д. . Смотри так-же */etc/issue*.

```
# uptime                                    # Аптайм, время прошедшее с момента запуска
системы + текущая нагрузка(LA)
# hostname                                  # Имя хоста
# hostname -i                               # IP адрес хоста (только для Linux)
# man hier                                  # Документация(man page) по иерархии
файловой системы
# last reboot                               # История перезагрузок(reboot)
```

Конфигурация железа

Железо определенное ядром при загрузке

```
# dmesg                                     # Железо, определенное при загрузке
# lsdev                                     # Информация об установлено железе
# dd if=/dev/mem bs=1k skip=768 count=256 | strings -n 30 # Прочитать данные из BIOS
```

Linux

```
# cat /proc/cpuinfo                         # Модель процессора
# cat /proc/meminfo                         # Физическая память
# grep MemTotal /proc/meminfo               # Объем физической памяти
# watch -n1 'cat /proc/interrupts'          # Следить за прерываниями
# free -m                                   # Использованная и свободная память (-m для
MB)
# cat /proc/devices                         # Сконфигурированные устройства
# lspci -tv                                 # Устройства PCI
# lsusb -tv                                 # Устройства USB
# lshal                                     # Показать список всех устройств с их
свойствами
# dmidecode                                 # Показать информацию из DMI/SMBIOS:, из
BIOS
```

FreeBSD

```
# sysctl hw.model                           # Модель процессора
# sysctl hw                                 # Большой список инфы о железе, ветка
переменных hw
# sysctl vm                                 # Информация по использованию памяти
# dmesg | grep usable memory                # Объем физической памяти
# sysctl -a | grep mem                      # _азличная информация о памяти ядра
# sysctl dev                                # Сконфигурированные устройства
# pciconf -l -cv                            # Устройства PCI
# usbdevs -v                                # Устройства USB
# atacontrol list                           # Устройства ATA
# camcontrol devlist -v                     # Устройства SCSI
```

Нагрузка, статистика, сообщения

Следующие команды применяются для мониторинга текущего состояния системы

```
# top                                       # Утилита TOP, различная системная
информация, процессы, LA
# mpstat 1                                  # Статистика касающаяся процессора
# vmstat 2                                  # Статистика виртуальной памяти, дисков и
процессора
# iostat 2                                  # Статистика операций ввода/вывода,I/O
# systat -vmstat 1                          # BSD суммарная системная статистика
# systat -tcp 1                             # BSD статистика tcp (так-же можно -ip)
# systat -netstat 1                         # BSD активные сетевые соединения
# systat -ifstat 1                          # BSD сетевой трафик на активных интерфейсах
# systat -iostat 1                          # BSD работа процессора и дисков
# tail -n 500 /var/log/messages             # Последние 500 сообщений Syslog из файла
messeges
# tail /var/log/warn                        # Системные предупреждения
```

Пользователи

```
# id                                        # Показать uid(имя), gid(группу), текущего
пользователя
```

```
# last                                    # Статистика последних входов в систему
# who                                     # Показать кто в системе в данный момент
# groupadd admin                          # Создать группу "admin" и пользователя
colin (Linux/Solaris)
# useradd -c "Colin Barschel" -g admin -m colin
# usermod -a -G             # Добавить существующего пользователя в группу (Debian)
# groupmod -A               # Добавить существующего пользователя в группу (SuSE)
# userdel colin                           # Удалить пользователя colin (Linux/Solaris)
# adduser joe                             # FreeBSD добавить пользователя joe
(интерактивно)
# rmuser joe                              # FreeBSD удалить пользователя joe
(интерактивно)
# pw groupadd admin                       # FreeBSD создать группу, используя утилиту
pw
# pw groupmod admin -m newmember          # FreeBSD добавить нового участника в группу
# pw useradd colin -c "Colin Barschel" -g admin -m -s /bin/tcsh # FreeBSD создать
пользователя (утилита pw)
# pw userdel colin; pw groupdel admin     # FreeBSD удалить пользователя и группу
(утилита pw)
```

Пароли, хранятся в файле */etc/shadow*, для Linux и в */etc/master.passwd* для операционной системы FreeBSD, в зашифрованном виде. Если файл *master.passwd* был изменен вручную, нужно воспользоваться командой *pwd_mkdb -p master.passwd* , что-бы пересобрать базу данных.

Что-бы временно запретить вход в систему, всем кроме root, используйте *nologin*.

```
# echo "Sorry no login now" > /etc/nologin           # (Linux)
# echo "Sorry no login now" > /var/run/nologin       # (FreeBSD)
```

Системные лимиты

Некоторые приложения (Proxy, Web сервера, базы данных) используют большие количества открытых файлов и сокетов, и как правило, установок по-умолчанию им недостаточно.

Linux

Оболочка/скрипт

За лимиты оболочки отвечает *ulimit*. Текущее состояние можно проверить *ulimit -a*. Например, что-бы увеличить кол-во открытых файлов с 1024 до 10240 нужно сделать:

```
# ulimit -n 10240                    # Команда верна только в оболочке
```

Так-же команда *ulimit* используется в скрипте, что-бы увеличить лимиты только для скрипта.

Пользователь/процесс

Лимиты пользователей и процессов устанавливаются в */etc/security/limits.conf*. Например:

```
# cat /etc/security/limits.conf
*   hard    nproc   250             # Лимит пользовательских процессов
asterisk hard nofile 409600         # Лимит открытых файлов для приложения
```

Общесистемные

Общесистемные лимиты устанавливаются командой *Sysctl*. Большинство этих переменных, действуют до перезагрузки, что-бы ограничения остались после ребута, внесите их в файл */etc/sysctl.conf*.

```
# sysctl -a                          # Показать все системные переменные
# sysctl fs.file-max                 # Показать максимально-возможное кол-во открытых
файлов
# sysctl fs.file-max=102400          # Изменить максимальное кол-во открытых файлов
# cat /etc/sysctl.conf
fs.file-max=102400                   # Постоянное значение в файле sysctl.conf
# cat /proc/sys/fs/file-nr           # Кол-во используемых файловых дескрипторов
```

FreeBSD

Оболочка/скрипт

Используйте команду *limits* в *csh* или *tcsh*.

пользователь/процесс

Лимиты по-умолчанию, устанавливаются при входе в систему в файле */etc/login.conf*. Неограниченные значение, ограничиваются общесистемными лимитами.

Общесистемные

Во FreeBSD, общесистемные лимиты регулируются так-же как в Linux, командой *Sysctl*. Постоянные ограничения устанавливаются в файлах */etc/sysctl.conf* и */boot/loader.conf*. Синтаксис записи аналогичен Linux, но ключи отличаются.

```
# sysctl -a                          # Показать все системные переменные
# sysctl kern.maxfiles=XXXX          # Максимальное кол-во файловых дескрипторов
kern.ipc.nmbclusters=32768           # Постоянные значения в /etc/sysctl.conf
kern.maxfiles=65536                  # типичные значения для Squid
kern.maxfilesperproc=32768
kern.ipc.somaxconn=4096              # TCP очередь, например для apache/sendmail
# sysctl kern.openfiles              # Кол-во используемых файловых дескрипторов
# sysctl kern.ipc.numopensockets     # Кол-во используемых сокетов
# sysctl -w net.inet.ip.portrange.last=50000 # Верхнее значение диапазона портов, по-
умолчанию: 1024-5000
# netstat -m                         # Статистика сетевых буферов памяти
```

Подробнее, смотрите Tuning Kernel Limits.

Solaris

Следующие значения в */etc/system* увеличат максимальное значение файловых дескрипторов на процесс:

```
set rlim_fd_max = 4096               # Жесткий лимит файловых дескрипторов на один
процесс
set rlim_fd_cur = 1024               # Мягкий лимит файловых дескрипторов на один
процесс
```

Runlevel — Режим работы системы / уровень запуска

Linux

Загрузившись, ядро стартует процесс *init*, который запускает *rc*, коорый в свою очередь выполняет все скрипты, соответствующего уровня запуска. Скрипты расположены в */etc/init.d* и слинкованы в */etc/rc.d/rcN.d*, где N, означает уровень запуска.

Уровень запуска по-умолчанию, установлен в */etc/inittab*, и как правило имеет значение 3 или 5:

```
# grep default: /etc/inittab
id:3:initdefault:
```

Текущий режим работы может быть изменен с помощью все того-же *init*. Например, перейдем с 3 уровня на 5:

```
# init 5                                # Переходим в режим 5
```

- 0 Shutdown and halt
- 1 Single-User mode (also S)
- 2 Multi-user without network
- 3 Multi-user with network
- 5 Multi-user with X
- 6 Reboot

Используйте *chkconfig* для конфигурирования программ, которые должны стартовать при загрузке в соответствующий режим.

```
# chkconfig --list                    # Список всех init-скриптов
# chkconfig --list sshd               # Показать статус sshd
# chkconfig sshd --level 35 on        # Конфигурирование sshd для уровня 3 и 5
# chkconfig sshd off                  # Отключение sshd для всех уровней
```

Debian и основанные на нем дистрибутивы, Ubuntu или Knoppix, для управления скриптами *Runlevel*, используют команду *update-rc.d*. По-умолчанию, 2, 3, 4 и 5 уровень для старта, и 0, 1 и 6 для останова.

```
# update-rc.d sshd defaults           # Активировать sshd с уровнем запуска по-
умолчанию
# update-rc.d sshd start 20 2 3 4 5 . stop 20 0 1 6 .  # Непосредственное указание
уровней запуска и останова
# update-rc.d -f sshd remove          # Запретить sshd для всех уровней
# shutdown -h now (or # poweroff)     # Остановить и выключить систему
```

FreeBSD

FreeBSD использует другой подход к процессу загрузки, нежели *SysV* (Linux, etc). Последний этап загрузки (*Single user* с *XServer* или без него), настроен в */etc/ttys*. Все скрипты расположены в */etc/rc.d/* и в */usr/local/etc/rc.d/* для сторонних производителей ПО. Запуск системных сервисов настраивается в */etc/rc.conf* и */etc/rc.conf.local*. Поведение по-умолчанию, задано в */etc/defaults/rc.conf*. Скрипты реагируют на *start* | *stop* | *status*.

```
# /etc/rc.d/sshd status
sshd is running as pid 552.
# shutdown now                        # Перключиться в однопользовательский режим
# exit                                # Вернуться в многопользовательский режим
# shutdown -p now                     # Выключить систему
# shutdown -r now                     # Перезагрузить
```

Процесс *init* может быть использован для управления установленным состоянием уровня. Например *init 6*, перезагрузит систему.

- 0 Остановить систему и выключить (сигнал USR2)
- 1 Перейти в Single user (сигнал TERM)
- 6 Перезагрузить машину (сигнал INT)
- c Блокировать последующие входы в систему (сигнал TSTP)

- q Перечитать ttys(5) файл (сигнал HUP)

Сброс пароля root

Linux, способ раз

В загрузчике (*lilo* или *grub*), введите следующие опции:

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

Ядро смонтирует корневую файловую систему, *init* запустит системный шелл (*bash*) Используйте команду *passwd*, что-бы изменить пароль *root* и перезагрузитесь. Если после загрузки, корневой раздел файловой системы смонтировался в режиме *read only* (только чтение), перемонтируйте его в *rw*, для чтения/записи:

```
# mount -o remount,rw /
# passwd                                # или удалите пароль root (/etc/shadow)
# sync; mount -o remount,ro /           # синхронизация перед монтированием в read only
# reboot
```

FreeBSD, способ раз

На экране загрузчика выберите *Single user* (опция 4), перемонтируйте корневую файловую систему в *rw*, используя утилиту *passwd* установите новый пароль для *root*.

```
# mount -u /; mount -a                  # перемонтирует корневую fs в режим rw
# passwd
# reboot
```

Unix и FreeBSD и Linux, способ два

Некоторые Unix системы, могут не дать вам исполнить вышеописанные трюки. Решение проблемы в том, что-бы смонтировать корневую файловую систему из другой OS (типа загрузочного CD) и сменить пароль.

- Загрузитесь с LiveCD или установочного диска в режиме восстановления.
- Найдите корневой партишн с помощью fdisk, например fdisk /dev/sda
- Смонтируйте его и используйте chroot:

```
# mount -o rw /dev/ad4s3a /mnt
# chroot /mnt                           # сдкелать /mnt корневой файловой системой
# passwd
# reboot
```

Сборка ядра (компиляция)

Linux

```
# cd /usr/src/linux
# make mrproper                         # Очистка, включая файлы конфигурации
# make oldconfig                        # Повторно использовать старые конфиги, если
есть
# make menuconfig                       # xconfig (Qt) или gconfig (GTK)
# make                                  # Создание сжатого образа ядра
# make modules                          # Компиляция модулей
```

```
# make modules_install                # Установка модулей
# make install                        # Установка ядра
# reboot
```

FreeBSD

При необходимости, обновите исходники системы (в */usr/src*) *csup*, для FreeBSD 6.2 и выше:

```
# csup
```

Пример конфига для *csup*:

```
*default host=cvsup2.ru.FreeBSD.org
*default base=/var/db
*default prefix=/usr
*default release=cvs tag=RELENG_7
*default delete use-rel-suffix
src-all
```

Что-бы изменить настройки ядра, скопируйте конфигурационный файл *GENERIC* под други именем и отредактируйте его под себя.

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC MYKERNEL
# cd /usr/src
# make buildkernel KERNCONF=MYKERNEL
# make installkernel KERNCONF=MYKERNEL
```

Переустановка всей операционной системы:

```
# make buildworld                          # Собрать мир (мир — все что не ядро)
# make buildkernel KERNCONF=MYKERNEL       # Сборка ядра (построение), как делали ранее
# make installkernel KERNCONF=MYKERNEL     # Установить ядро
# reboot
# mergemaster -p                           # Сравнить только необходимые уонфиги
# make installworld                        # Установить мир
# mergemaster -i -U                        # Обновить все конфигурационные и другие
файлы
# reboot
```

При незначительных изменениях в исходных текстах, можно использовать ключ *NO_CLEAN=yes*, что-бы избежать пересборку всего дерева исходников.

```
# make buildworld NO_CLEAN=yes       # Не удалять старые объектные файлы
# make buildkernel KERNCONF=MYKERNEL NO_CLEAN=yes
```

Модули ядра

Linux

```
# lsmod                              # Показать все модули загруженные в ядро
# modprobe isdn                      # Загрузить модуль (в данном случае isdn)
```

FreeBSD

```
# kldstat                            # Списоз загруженных модулей
# kldload crypto                     # Загрузить модуль (здесь crypto)
```

Восстановление загрузчика grub

```
# mount /dev/sda6 /mnt                # монтировать Linux partion на /mnt
# mount --bind /proc /mnt/proc        # монтировать подсистему proc в /mnt
# mount --bind /dev /mnt/dev          # монтировать устройства в /mnt
# chroot /mnt                         # сменить корневую директрорию на Linux partition
# grub-install /dev/sda               # переустановить grub со старыми опциями
```

Шифрование разделов

Linux

В данном руководстве используется *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.

Шифрование разделов диска с помощью LUKS

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*, если насчет *LUKS* ничего не появилось, читайте ниже "dm-crypt без LUKS". Для начала создайте раздел, если необходимо *fdisk /dev/sdc*.

Как создать зашифрованный 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
```

Монтировать

```
# cryptsetup luksOpen /dev/sdc1 sdc1
# mount -t ext3 /dev/mapper/sdc1 /mnt
```

Размонтировать

```
# umount /mnt
# cryptsetup luksClose sdc1
```

dm-crypt без LUKS

```
# 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
```

Делайте 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 по новой.

FreeBSD

Пара популярных модулей для шифрования дисков в операционной системе FreeBSD, это *gbde* и *geli*. *Geli* более быстрый т.к использует аппаратное ускорение. Смотрите FreeBSD handbook Chapter 18.6 для более подробного описания. Для работы, *geli* должен быть загружен как модуль ядра, или встроен в него на стадии компиляции.

```
options GEOM_ELI
device crypto                                       # Или загрузить в качестве модуля
ядра:


# echo 'geom_eli_load="YES"' >> /boot/loader.conf   # Или kldload geom_eli
```

Использование пароля и ключа

Автор пользуется данными настройками для типичного шифрования разделов, он использует пароль и ключ для шифрования "*Master key* — основного ключа". Что-бы смонтировать зашифрованный раздел, понадобится и пароль и ключ */root/ad1.key*. "*Master key*" хранится вутри раздела и невидим. Следующий пример типичен для USB или файлового образа.

Создаем шифрованный раздел

```
# dd if=/dev/random of=/root/ad1.key bs=64 count=1  # Этот ключ шифрует Master key
# geli init -s 4096 -K /root/ad1.key /dev/ad1       # -s 8192 и ОК для дисков
# geli attach -k /root/ad1.key /dev/ad1             # DO создает резервную копию
/root/ad1.key
# dd if=/dev/random of=/dev/ad1.eli bs=1m           # Опционально и занимает много
времени
# newfs /dev/ad1.eli                                # Создать файловую систему
# mount /dev/ad1.eli /mnt                           # Монтирование шифрованного
раздела
```

Attach

```
# geli attach -k /root/ad1.key /dev/ad1
# fsck -ny -t ffs /dev/ad1.eli                      # Если есть сомнения, проверьте
файловую систему
# mount /dev/ad1.eli /mnt
```

Detach

Процедура размонтирования производится автоматически при выключении.

```
# umount /mnt
# geli detach /dev/ad1.eli
```

/etc/fstab

Монтирование шифрованного раздела можно сконфигурировать через */etc/fstab*. Пароль будет запрошен при загрузке.

```
# grep geli /etc/rc.conf
geli_devices="ad1"
geli_ad1_flags="-k /root/ad1.key"
# grep geli /etc/fstab
/dev/ad1.eli         /home/private              ufs             rw      0       0
```

Только по паролю

Это более подходящий способ для шифрования флэшки или образа на основе файла, запрашивается только пароль. В данном случае не нужно волноваться о файлах ключей. Процедура напоминает вышеописанную, за исключением создания файлов ключей. Зашифруем образ размером 1 Гб, созданный из файла */cryptedfile*.

```
# dd if=/dev/zero of=/cryptedfile bs=1M count=1000  # Создаем 1Гб файл
# mdconfig -at vnode -f /cryptedfile
# geli init /dev/md0                                # Зашифровать только по паролю
# geli attach /dev/md0
# newfs -U -m 0 /dev/md0.eli
# mount /dev/md0.eli /mnt
# umount /dev/md0.eli
# geli detach md0.eli
```

Теперь этот образ можно примонтировать на другую машину, просто введя пароль.

```
# mdconfig -at vnode -f /cryptedfile
# geli attach /dev/md0
# mount /dev/md0.eli /mnt
```

Процессы

Каждая запущеная программа в операциооных системах Unix (и не только), имеет уникальный номер, *PID* процесса. Список всех запущенных процессов можно получить утилитой *ps*.

```
# ps -auxefw                          # Полный список всех запущеных процессов
```

Однако, зачастую, целесообразней использовать эту команду с выводом на *pgrep*:

```
# ps axww | grep cron
586  ??  Is     0:01.48 /usr/sbin/cron -s
# ps axjf                             # Все процессы в виде дерева (Linux)
# ps aux | grep 'ss[h]'               # Найти все PID процесса ssh, исключая PID
процесса grep
# pgrep -l sshd                       # Найти PID процесса по имени(или по его части)
# echo $$                             # PID процесса текушей оболочки
# fuser -va 22/tcp                    # Список процессов использующих порт 22 (Linux)
# pmap PID                            # Карта памяти процессов (выявление утечек
памяти) (Linux)
# fuser -va /home                     # Список процессов, имеющих доступ в раздел
/home
# strace df                           # Отслеживать(Trace) системные вызовы и сигналы
# truss df                            # Тоже что и выше, но на
FreeBSD/Solaris/Unixware
```

Приоритет процесса

Изменить приоритет процесса можно командой *renice*. Отрицательное значение, означает более высокий приоритет.

```
# renice -5 586                       # Повысить приоритет процесса с PID 586
586: old priority 0, new priority -5
```

С помощью *nice* можно запускать процессы с определенным приоритетом. Что-бы узнать, используете вы, */usr/bin/nice* или *nice* встроенный в шелл, используйте *which nice*).

```
# nice -n -5 top                      # Повысить приоритет процесса(/usr/bin/nice)
# nice -n 5 top                       # Понизить приоритет процесса(/usr/bin/nice)
# nice +5 top                         # nice Встроенный в шелл, понизить приоритет
процесса
```

В то время как *nice* отвечает за распределение процессорного времени, другая полезная команда *ionice*, распределяет дисковый *IO*. Она весьма полезна в случае приложений, активно использующих дисковый *IO* (например компилирование). Вы можете установить для приложения соответствующий класс (*idle — best effort — real time*), загляните в man, там все довольно доступно рассказано.

```
# ionice c3 -p123                     # Установить класс idle для pid 123 (только
Linux)
# ionice -c2 -n0 firefox              # Запустить Фаерфокс с высоким приоритетом и
классом best effort
# ionice -c3 -p$$                     # Установить для текущей оболочки класс idla
```

Последняя команда очень полезна при компилировании или отладке больших проектов. Любая команда, запущенная из текущей оболочки будет иметь пониженный приоритет. Переменная $ $, содержит *PID* текушей оболочки (попробуйте *echo $$*)
FreeBSD использует *idprio/rtprio* (0 = максимальный приоритет, 31 = наиболее свободный(*most idle*)):

```
# idprio 31 make                      # Компилировать с низким приоритетом
# idprio 31 -1234                     # Установить низкий приоритет для PID 1234
# idprio -t -1234                     # -t удаляет real time/idle приоритеты
```

background/foreground

Процессы запущенные в шелле, можно переключать в фоновый режим (*background*), приостанавливать, нажав [Ctrl]-[Z], и выводить обратно в *foreground*, так-же можно использовать команды *bg* and *fg*. Команда *jobs*, выведет список запущенных в фоновом режиме процессов.

```
# ping cb.vu > ping.log
^Z                                    # Команда ping приостановлена [Ctrl]-[Z]
# bg                                  # Отправить в фон и продолжать выполнение
# jobs -l                             # Список фоновых процессов
[1]  - 36232 Running                        ping cb.vu > ping.log
[2]  + 36233 Suspended (tty output)         top
# fg %2                               # Вернуть процесс 2 в обычный режим
```

Что-бы процесс продолжал свое выполнение после закрытия оболочки, можно воспользоваться командой *nohup*.

```
# nohup ping -i 60 > ping.log &
```

Программа top

Программа *top* показывает работчие данные по запущенным процессфм. Так-же обратите внимание на программу *htop*, это более расширенная версия программы, работает на Linux и FreeBSD (*/usr/ports/sysutils/htop*). Что-бы получить справку по ключам, во время работы *top*, нажмите h. Вот некоторые полезные ключи:

- u [имя пользователя] — Посмотреть процессы принадлежащие конкретному пользователю. Используйте + или пробел для возврата в режим полного просмотра.
- k [pid] — Убить (завершить) процесс с *pid*.
- 1 — Показать статистику по процессору (только Linux)
- R — Переключить сортировку

Команда kill

Остановка процессов или отправка сигнала с помощью команд *kill* или *killall*.

```
# ping -i 60 cb.vu > ping.log &
[1] 4712
# kill -s TERM 4712                  # то-же что и kill -15 4712
# killall -1 httpd                   # отправить сигнал HUP процессу с точным именем
# pkill -9 http                      # отправить сигнал TERM процессу с именем или
его частью
# pkill -TERM -u www                 # отправить сигнал TERM процессу выполняющемуся
от имени конкретного пользователя
# fuser -k -TERM -m /home            # убить процессы имеющие доступ к /home
```

Наиболее важные сигналы:

- 1 HUP — Часто используется для перечитывания конфигурационных файлов
- 2 INT — Прервать(interrupt)
- 3 QUIT — Выйти
- 9 KILL — Безусловно прибить процесс
- 15 TERM — Мягкое прерывание

Файловая система

Права доступа к файлам и каталогам

Менять права доступа и владельца файлов и каталогов в Unix подобных операционных системах, можно с помощью команд *chmod* и *chown*. Маску для установки прав на создаваемые файлы, можно изменить глобально, в */etc/profile* для Linux и в */etc/login.conf* для FreeBSD. Обычно, маска по-умолчанию 022. Значение *umask* вычитается из 777, таким образом права доступа будут иметь значение 755.

```
exec — разрешено выполнение
read — право на чтение
write — право на запись
SUID bit — атрибут файла, в совокупности с атрибутом исполняемого файла, позволяет
запускаемому файлу выполняться с эффективным UID владельца файла, а не того, кто
запускает файл
1 --x execute                        # Права 764 = exec/read/write | read/write |
read
2 -w- write                          # Для:       |--  Owner  --|   |- Group-|   |
Oth|
4 r-- read
ugo=a                              u=user, g=group, o=others, a=everyone
```

```
# chmod [OPTION] MODE[,MODE] FILE    # MODE имеет форму: [ugoa]*([-+=]([rwxXst]))
# chmod 640 /var/log/maillog         # Установить права доступа равными -rw-r-----
# chmod u=rw,g=r,o= /var/log/maillog # Как и выше
# chmod -R o-r /home/*               # _екурсивно изменить права, запретить чтение
для Other
# chmod u+s /path/to/prog            # Установить SUID бит на исполняемый файл (тут
осторожней, вы должны понимать, что вы делаете)
# find / -perm -u+s -print           # Найти все программы с установленным SUID битом
# chown user:group /path/to/file     # Установить пользователя и группу как владеющих
файлом
# chgrp group /path/to/file          # Изменить группу владеющую файлом
# chmod 640 `find ./ -type f -print` # Изменить права доступа на 640 для всех файлов
# chmod 751 `find ./ -type d -print` # Изменить права доступа на 751 для всех
директорий
```

Информация о дисках

```
# diskinfo -v /dev/ad2               # Посмотреть информацию о диске (sector/size)
FreeBSD
# hdparm -I /dev/sda                 # Информация о IDE/ATA диске (Linux)
# fdisk /dev/ad2                     # Показать изменить разделы диска
# smartctl -a /dev/ad2               # Показать SMARTинформацию диска
```

Загрузка

FreeBSD

Что-бы загрузить старое ядро, в аварийной ситуации, например после неудачной сборки и установки нового, остановите загрузку, нажав 6 во время обратного отсчета, что-бы попасть в приглашение командной строки.

```
# unload
# load kernel.old
# boot
```

Точки монтирования, использование дисков

```
# mount | column -t                  # Показать смонтированные файловые системы
# df                                 # Показать кол-во свободного места и
смонтированные устройства
# cat /proc/partitions               # Показать все зарегистрированные разделы
(Linux)
```

Информация о директориях

```
# du -sh *                           # _азмеры директорий в виде списка
# du -csh                            # Суммарный объем текущей директории
# du -ks * | sort -n -r              # Список директорий, отсортированный по объему в
килобайтах
# ls -lSr                            # Список директорий, обратная сортировка
```

Кто какие файлы открыл

Иногда необходимо выяснить, какой файл заблокировал раздел, из-за чего команда *umount* выдает соответствующую ошибку.

```
# umount /home/
umount: unmount of /home             # _азмонтировать раздел невозможно, пока /home
заблокирован
failed: Device busy
```

FreeBSD и большинство Unix подобных систем

```
# fstat -f /home                     # для точки монтирования
# fstat -p PID                       # для приложения с PID
# fstat -u user                      # для имени пользователя
```

Найти открытый файл 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
```

Найти файл с *inum* 212042 в директории */var* можно так:

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

Linux

Найти открытый файл в директории с помощью *fuser* или *lsof*:

```
# fuser -m /home                     # Список процессов имеющих доступ к /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)
```

Найти по *PID* приложения:

```
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
```

По имени файла:

```
# 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
```

Монтирование/перемонтирование файловых систем

Например *cdrom*, прописанный в */etc/fstab*:

```
# mount /cdrom
```

Или можно найти устройство в */dev* или в выводе *dmesg*

FreeBSD

```
# mount -v -t cd9660 /dev/cd0c /mnt  # Монтирование диска Cdrom (способ первый)
# mount_cd9660 /dev/wcd0c /cdrom     # Монтирование диска Cdrom (способ второй)
# mount -v -t msdos /dev/fd0c /mnt   # Дискета
```

Запись в */etc/fstab*:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/acd0               /cdrom          cd9660  ro,noauto       0       0
```

Разрешить пользователям монтирование дисков:

```
# sysctl vfs.usermount=1  # Или впишите строку "vfs.usermount=1" in /etc/sysctl.conf
```

Linux

```
# mount -t auto /dev/cdrom /mnt/cdrom   # Типичная команда монтирования диска cdrom
# mount /dev/hdc -t iso9660 -r /cdrom   # Монтирование диска IDE
# mount /dev/scd0 -t iso9660 -r /cdrom  # Монтирование диска SCSI cdrom
# mount /dev/sdc0 -t ntfs-3g /windows   # Монтирование диска SCSI
```

Запись в */etc/fstab*:

```
/dev/cdrom   /media/cdrom  subfs noauto,fs=cdfss,ro,procuid,nosuid,nodev,exec 0 0
```

Монтирование FreeBSD раздела с Linux

Посмотрите номер раздела в *fdisk*, обычно это корнеыой раздел, но может быть и на другом *BSD* слайсе. Если на разделе FreeBSD много слайсов, их не будет видно через *fdisk*, но их можно найти в */dev/sda** или */dev/hda**.

```
# fdisk /dev/sda                      # Найти FreeBSD раздел
/dev/sda3   *        5357        7905    20474842+  a5  FreeBSD
# mount -t ufs -o ufstype=ufs2,ro /dev/sda3 /mnt
/dev/sda10 = /tmp; /dev/sda11 /usr   # Другой слайс
```

Перемонтирование

Перемонтировать устройство без предварительного размонтирования, например для *fsck*

```
# mount -o remount,ro /             # Linux
# mount -o ro /                     # FreeBSD
```

Копировать поток данных с *CDROM*'а в файл *ISO образа*.

```
# dd if=/dev/cd0c of=file.iso
```

Создание swap раздела на лету

Предположим вам нужно увеличить swap раздел, скажем до 2 гигабайт, */swap2gb* (для Linux)

```
# dd if=/dev/zero of=/swap2gb bs=1024k count=2000
# mkswap /swap2gb                   # Создать swap
```

```
# swapon /swap2gb                    # Включить swap, теперь его можно использовать
# swapoff /swap2gb                   # Отключить swap
# rm /swap2gb
```

Монтирование SMB раздела

CIFS — Common Internet File System
SMB — server message block

Предположим вам нужно получить доступ на расшаренному *SMB* разделу *myshare* на сервере *smbserver*, адрес набираемый на Windows машине будет *\\smbserver\myshare*. Монтировать будем на */mnt/smbshare*. Не забывайте, для *cifs* требуется IP адрес или доменное имя.

Linux

```
# smbclient -U user -I 192.168.16.229 -L //smbshare/    # List the shares
# mount -t smbfs -o username=winuser //smbserver/myshare /mnt/smbshare
# mount -t cifs -o username=winuser,password=winpwd //192.168.16.229/myshare
/mnt/share
```

Кроме того пакет *mount.cifs* позволяет хранить привилегии в файле, например */home/user/.smb*:

```
username=winuser
password=winpwd
```

И теперь монтируем:

```
# mount -t cifs -o credentials=/home/user/.smb //192.168.16.229/myshare /mnt/smbshare
```

FreeBSD

Используйте ключ -I, что-бы задать IP адрес (или *DNS*); *smbserver*, это Windows имя.

```
# smbutil view -I 192.168.16.229 //winuser@smbserver    # Список расшаренных ресурсов
# mount_smbfs -I 192.168.16.229 //winuser@smbserver/myshare /mnt/smbshare
```

Монтировать образ

Linux loop-back

```
# mount -t iso9660 -o loop file.iso /mnt               # Монтировать образ CD
# mount -t ext3 -o loop file.img /mnt                  # Монтировать образ с
файловой системой ext3
```

FreeBSD

Используя *md* — устройство памяти (если нужно, сделайте *kldload md.ko*):

```
# mdconfig -a -t vnode -f file.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
# umount /mnt; mdconfig -d -u 0                        # Очистить устройство памяти
```

Или используя псевдоустройство (*VN, Virtual node*):

```
# vnconfig /dev/vn0c file.iso; mount -t cd9660 /dev/vn0c /mnt
# umount /mnt; vnconfig -u /dev/vn0c                   # Очистить псевдоустройство
```

Создание и запись образа ISO

Будем копировать cd или dvd сектор за сектором.

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

Используйте *mkisofs* что-бы создать образ из файла в директории. Для преодоления ограничений имен файлов используйте опцию -r, включающую расширение *RockRidge*, основное для UNIX систем, -J включает *Joliet*, используемое Microsoft, -L разрешает *ISO9660* имена, начинающиеся точкой.

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

Во FreeBSD, *mkisofs* можно установить из портов /usr/ports/sysutils/cdrtools.

Запись CD/DVD ISO образов

FreeBSD

FreeBSD не устанавливает *DMA* на *ATAPI* устройства, это можно сделать через переменную *sysctl* или в файле */boot/loader.conf*, следующими записями.

```
hw.ata.ata_dma="1"
hw.ata.atapi_dma="1"
```

Используйте *burncd* для ATAPI устройств (*burncd*, стандартная программа, часть базовой системы) и *cdrecord* (из */usr/ports/sysutils/cdrtools*) для SCSI устройств.

```
# burncd -f /dev/acd0 data imagefile.iso fixate    # Для ATAPI устройств
# cdrecord -scanbus                                # Найти рекордер
# cdrecord dev=1,0,0 imagefile.iso
```

Linux

Так-же используйе *cdrecord*, как описано выше. Кроме того можно использовать родной ATAPI интерфейс:

```
# cdrecord dev=ATAPI -scanbus
```

Записывайте, как было описано выше.

dvd+rw-tools

Пакет dvd+rw-tools (FreeBSD: ports/sysutils/dvd+rw-tools) имеет весь функционал необходимый для работы с DVD, плюс *growisofs*, для записи CD или DVD. Документацию с примерами можно найти в FreeBSD handbook Глава 18.7

```
# -dvd-compat закрывает диск
# growisofs -dvd-compat -Z /dev/dvd=imagefile.iso     # Записать существующий iso
образ
# growisofs -dvd-compat -Z /dev/dvd -J -R /p/to/data  # Записать напрямую
```

Конвертировать образ из Nero .nrg файла в файл .iso

Nero добавляет к образу заголовок в 300кб, его можно обрезать с помощью *dd*.

```
# dd bs=1k if=imagefile.nrg of=imagefile.iso skip=300
```

Конвертировать образ bin/cue в .iso

Это можно сделать с помощью небольшой программы, bchunk. Во FreeBSD ее можно найти в портах */usr/ports/sysutils/bchunk*.

```
# bchunk imagefile.bin imagefile.cue imagefile.iso
```

Создание образа на основе файла

Например, раздел размером 1Гб использует файл */usr/vdisk.img*. В данном случае мы используем ключ -u 0, но номер может быть любым.

FreeBSD

```
# dd if=/dev/random of=/usr/vdisk.img bs=1K count=1M
# mdconfig -a -t vnode -f /usr/vdisk.img -u 0         # Создаем устройство /dev/md1
# bsdlabel -w /dev/md0
# newfs /dev/md0c
# mount /dev/md0c /mnt
# umount /mnt; mdconfig -d -u 0; rm /usr/vdisk.img    # Очистить md
```

Образ созданный из файла может быть смонтирован в процессе загрузки системы, путем записи строки в */etc/rc.conf* и */etc/fstab*.
Проверить правильность ваших настроек можно с помощью команды */etc/rc.d/mdconfig start* (предварительно удалив устройство *md0* с помощью команды *# mdconfig -d -u 0*).
Имейте в виду, что автоматическая монтирование образа будет работать, только если файл образа, лежит НЕ в корневом разделе, в силу того что скрипт */etc/rc.d/mdconfig* выполняется на ранней стадии загрузки, когда корневой раздел еще не доступен на запись. Образы расположенные вне корневого раздела будут смонтированы позже, скриптом */etc/rc.d/mdconfig2*.

> /boot/loader.conf:
> md_load="YES"
>
> /etc/rc.conf:
> mdconfig_md0="-t vnode -f /usr/vdisk.img" # */usr* не в корневом разделе
>
> /etc/fstab: (0 0 в конце, очень важны, это укажет *fsck* игнорировать проверку устройства, так как оно еще не существует)
> /dev/md0 /usr/vdisk ufs rw 0 0

Кроме того, в последствии можно увеличить размер образа, скажем на 300 мб.

```
# umount /mnt; mdconfig -d -u 0
# dd if=/dev/zero bs=1m count=300 >> /usr/vdisk.img
# mdconfig -a -t vnode -f /usr/vdisk.img -u 0
# growfs /dev/md0
# mount /dev/md0c /mnt                                # Теперь файловый раздел на 300
мб больше
```

Linux

```
# 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                          # Очистить
```

Linux и losetup

/dev/zero намного быстрей, чем *urandom*, но менее защищенн для шифрования.

```
# dd if=/dev/urandom of=/usr/vdisk.img bs=1024k count=1024
# losetup /dev/loop0 /usr/vdisk.img                       # Создать /dev/loop0
# mkfs.ext3 /dev/loop0
# mount /dev/loop0 /mnt
# losetup -a                                              # Проверить
# umount /mnt
# losetup -d /dev/loop0                                   # Отсоединить
# rm /usr/vdisk.img
```

Создание файловой системы в памяти

Файловая система в памяти очень быстрая, имеет смысл использовать ее для приложений с высоким дисковым IO. Создадим раздел размером 64 мб и смонтируем его в */memdisk:*

FreeBSD

```
# mount_mfs -o rw -s 64M md /memdisk
# umount /memdisk; mdconfig -d -u 0                       # Очистить md устройство
md     /memdisk     mfs     rw,-s64M    0   0            # запись в /etc/fstab
```

Linux

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

Производительность дисков

Чтение и запись *1гб* файла в разделе *ad4s3c* (*/home*)

```
# time dd if=/dev/ad4s3c of=/dev/null bs=1024k count=1000
# time dd if=/dev/zero bs=1024k count=1000 of=/home/1Gb.file
# hdparm -tT /dev/hda                                        # Только Linux
```

Управление пакетами

Список установленных пакетов

```
# rpm -qa                           # Список установленных пакетов(RH, SuSE, RPM)
# dpkg -l                           # Debian, Ubuntu
# pkg_info                          # Список установленных пакетов во FreeBSD
# pkg_info -W smbd                  # Посмотреть, какому пакету принадлежит файл во
FreeBSD
# pkginfo                           # Solaris
```

Установка пакетов / удаление пакетов

Yast2/yast для *SuSE*, redhat-config-packages для *Red Hat.*

```
# rpm -i pkgname.rpm                  # Установить пакет (RH, SuSE, RPM)
# rpm -e pkgname                      # Удалить пакет
```

Debian

```
# 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
```

Gentoo

Gentoo использует *emerge* для управления системой своих пакетов.

```
# emerge --sync                       # Синхронизировать локальное дерево портов
# emerge -u packagename               # Установить пакет или обновить пакет
# emerge -C packagename               # Удалить пакет
# revdep-rebuild                      # Восстановить зависимости пакетов
```

Solaris

Путь к обчно выглядит так */cdrom/cdrom0*.

```
# pkgadd -d /Solaris_9/Product SUNWgtar
# pkgadd -d SUNWgtar                  # Добавить скачанный пакет(сначала распаковать
bunzip2)
# pkgrm SUNWgtar                      # Удалить пакет
FreeBSD
# pkg_add -r rsync                    # Скачать и установить пакет rsync.
# pkg_delete /var/db/pkg/rsync-xx     # Удалить пакет rsync
```

Установить, откуда будут получены пакеты, можно переменной *PACKAGESITE*. Например:

```
# export PACKAGESITE=ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages/Latest/
# или ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-6-stable/Latest/
```

Порты FreeBSD

Дерево портов в системе FreeBSD, это коллекция программ, готовых к компилированию и установке, располагается по адресу */usr/ports*. Обновить дерево портов можно с помощью программы *portsnap*.

```
# portsnap fetch extract              # Получить и разпаковать свежее дерево портов
(при первом запуске)
# portsnap fetch update               # Обновить дерево портов
# cd /usr/ports/net/rsync/            # Перейти в директорию порта для установки
# make install clean                  # Установить порт и очистить директорию
установки (смотрите man ports)
# make package                        # Создать из порта бинарный пакет
```

Пути к библиотекам

Библиотеки проверяются с помощью команды *ldd*, и управляются *ldconfig*.
На примере программы *rsync*:

```
# ldd /usr/bin/rsync                  # Список необходимых библиотек для rsync
```

```
# ldconfig -n /path/to/libs/          # Добавить путь к разделяемым библиотекам
# ldconfig -m /path/to/libs/          # FreeBSD
# LD_LIBRARY_PATH                     # Данная переменная устанавливает путь к
библиотекам
```

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*)Процедура создания SSL сертификата

- Клиент должен создать сертификат, со всеми необходимыми данными.
- Отправить запрос на сертификацию в один из "*центров сертификации*" (далее *ЦС*). Также на данном этапе, беднт создан приватный ключ на локальной машине.
- После обработки запроса, сертификат подписывается секретным ключем *ЦС*. Клиент имея публичный ключ *ЦС*, 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/ssl/openssl.cnf*, согласно вашей конфигурации.
Вот часть конфигурационного файла *openssl.cnf*, имеющая отношение к делу:

```
[ CA_default ]
dir             = /usr/local/certs/CA       # Где все хранить
certs           = $dir/certs                # Где хранить сертификаты
crl_dir         = $dir/crl                  # Где хранить списки отзыва сертификатов
(CRL)
database        = $dir/index.txt            # Индексный файл базы данных
```

Убедитесь что директории существуют, иначе создайте их.

```
# mkdir -p /usr/local/certs/CA
# cd /usr/local/certs/CA
# mkdir certs crl newcerts private
# echo "01" > serial                       # Только если нет порядкового номера
# touch index.txt
```

Если вы собираетесь получать подписанный сертификат от какого-либо *ЦС*, вам нужно отправить запрос на сертификацию (*CSR*). После обработки запроса, сертификат будет подписан на определенный срок (например 1 год).

Создать сертификат полномочий

Если у вас нет сертификата, подписанного ЦС, и вы не планируете отправлять запрос на сертификацию, можно создать свой сертификат.

```
# openssl req -new -x509 -days 730 -config /etc/ssl/openssl.cnf -keyout
CA/private/cakey.pem -out CA/cacert.pem
```

Запрос сертификации (CSR)

Если ваше приложение не поддерживает шифрование (например *UW-IMAP*), отключите его (шифрование), с помощью опции *-nodes*.

```
# 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
```

Сохраните созданный 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 *ЦС* сертификат является действующим.
ниже, замените "*servername*" на имя своего сервера

```
# cat newreq.pem newkey.pem > new.pem
# openssl ca -policy policy_anything -out servernamecert.pem -config
/etc/ssl/openssl.cnf -infiles new.pem
# mv newkey.pem servernamekey.pem
```

Ткперь *servernamekey.pem* — содержит приватный ключ а *servernamecert.pem* — сертификат сервера.

Создание объединенного сертификата

IMAP сервер желает иметь все приватные ключи и серверные сертификаты в одном файле, сделать это не сложно, но файл должен храниться в очень безопасном месте.
Создадим файл *servername.pem* содержащий и сертификаты и ключи.

- Открыть файл *servernamekey.pem* в текстовом редакторе и скопировать приватный ключ в файл *servername.pem*.
- Тоже самое нужно проделать с сертификатом *servernamecert.pem*.

Окончательный вариант файла *servername.pem*, будет выглядеть примерно так:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDutWy+o/XZ/[...]qK5LqQgT3c9dU6fcR+WuSs6aejdEDDqBRQ
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIERzCCA7CgAwIBAgIBBDANB[...]iG9w0BAQQFADCBxTELMAkGA1UEBhMCREUx
-----END CERTIFICATE-----
```

Что у нас теперь есть в директории */usr/local/certs/*:

- CA/private/cakey.pem (CA приватный ключ)
- CA/cacert.pem (CA публичный ключ)
- certs/servernamekey.pem (приватный ключ сервера)
- certs/servernamecert.pem (подписанный сертификат сервера)
- certs/servername.pem (сертификат сервера и приватный ключ)

Приватный ключ в безопасном месте!

Просмотр информации 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 cb.vu:443            # Проверить сертификат Веб сервера
```

Конвертирование форматов

Кодировки текста

Для конвертирования текстового файла из одной кодировки в другую, служит команда *iconv*.

```
# iconv -f  -t
# iconv -f ISO8859-1 -t UTF-8 -o file.input > file_utf8
# iconv -l                           # Список всех поддерживаемых кодировок
```

Без опции -f, *iconv* будет использовать локальную кодировку.

Символы новой строки Unix — DOS

Конвертирование символов новой строки *DOS (CR/LF)* в Unix формат и обратно. Смотрите также *dos2unix* и *unix2dos*.

```
# sed 's/.$//' dosfile.txt > unixfile.txt                  # DOS в UNIX
# awk '{sub(/\r$/,"");print}' dosfile.txt > unixfile.txt   # DOS в UNIX
# awk '{sub(/$/,"\r");print}' unixfile.txt > dosfile.txt   # UNIX в DOS
```

Конвертирование в Windows окружении, используя *sed* или *awk* из *mingw* или *cygwin*.

```
# sed -n p unixfile.txt > dosfile.txt
# awk 1 unixfile.txt > dosfile.txt   # UNIX в DOS (используя оболочку cygwin)
```

Конвертировать формата PDF в Jpeg и объединение PDF файлов

Конвертировать *PDF* в *JPG* (или *PNG*), можно с помощью *gs* (*GhostScript*), из каждой страницы будет создано отдельно изображение. Эту-же задачу можно решить с помощью *convert* (из *ImageMagick* или *GraphicsMagick*).

```
# gs -dBATCH -dNOPAUSE -sDEVICE=jpeg -r150 -dTextAlphaBits=4 -dGraphicsAlphaBits=4 \
-dMaxStripSize=8192 -sOutputFile=unixtoolbox_%d.jpg unixtoolbox.pdf
# convert example.pdf example-%03d.png
# convert *.jpeg images.pdf          # Создать простой PDF документ из всех картинок
```

Кроме того *Ghostscript* может объединить несколько *PDF* файлов в один большой файл.

```
# gs -q -sPAPERSIZE=a4 -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -sOutputFile=all.pdf \
file1.pdf file2.pdf ...              # В Windows используйте '#' вместо '='
```

Конвертировать форматов видео

Сжатие видео, кодеком mpeg4 с исправлением звука.

```
# mencoder -o videoout.avi -oac mp3lame -ovc lavc -srate 11025 \
```

```
-channels 1 -af-adv force=1 -lameopts preset=medium -lavcopts \
vcodec=msmpeg4v2:vbitrate=600 -mc 0 vidoein.AVI
```

Так-же смотрите sox.

Копирование audio CD

Сохранить трэки можно программой cdparanoia (FreeBSD порт */usr/ports/audio/cdparanoia/*), *oggenc* поможет конвертировать в формат *Ogg Vorbis*, *lame* конвертирует в *mp3*.

```
# cdparanoia -B                       # Копировать трэки в WAV файлы, в текущую
директорию.
# lame -b 256 in.wav out.mp3          # Кодировать в mp3 с битрейтом 256 kb/s
# for i in *.wav; do lame -b 256 $i `basename $i .wav`.mp3; done
# oggenc in.wav -b 256 out.ogg        # Кодировать в Ogg Vorbis 256 kb/s
```

Шифрование файлов

OpenSSL

Шифрование отдельного файла

```
# openssl aes-128-cbc -salt -in file -out file.aes    # зашифровать файл
# openssl aes-128-cbc -d -salt -in file.aes -out file   # расшифровать файл
```

Естественно файл может быть и архивом.

Архивирование и шифрование директории

```
# tar -cf - directory | openssl aes-128-cbc -salt -out directory.tar.aes      #
заархивировать и зашифровать директории
# openssl aes-128-cbc -d -salt -in directory.tar.aes | tar -x -f -           #
расшифровать директории и распаковать архив
```

То-же самое, только тип архива tar.gz

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

- Используйте *-k mysecretpassword* после aes-128-cbc, что-бы не спрашивался пароль, но имейте в виду, это очень не безопасно.
- Используйте aes-256-cbc вместо aes-128-cbc для получения более устойчивого шифра, увеличивается потребление процессора.

GPG шифрование

Альтернатива *PGP*, распространяемая по лицензии *GPL* (*GNU General Public License*).

GnuPG очень 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                         # _асшифровать файл (-o другой файл)
```

Шифрование с использованием пары ключей

Для более полной информации смотрите GPG Quick Start, GPG/PGP Basics и gnupg documentation.
Приватный ключ и публичный ключ, основа ассиметричной криптографии. О чем нужно помнить:

- Ваш публичный ключ используется другими для шифрования файлов, которые, как получатель, можете расшифровать только вы (даже не тот, кто его шифровал).
- 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.
- Ключевой файл, может содержать несколько ключей.

Сперва 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
```

Ключи сохраняются в *~/.gnupg/* в Unix подобных операционных системах и в *C:/Documents and Settings/%USERNAME%/Application Data/gnupg/*. в Windows.

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

Некоторые часто используемые 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                       # _асшифровать. Используется опция -o,
иначе пойдкт в stdout
```

Шифрование — расшифровка с использованием ключей

Для начала вам нужно экспортировать ваш публичный ключ, что-бы им могли пользоваться для расшифровки данных. Так-же вы 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 KEylD   # 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                     # _асшифровать файл, зашифрованный
Alice для вас.
```

Управление ключами

```
# gpg --list-keys                             # Список публичных ключей с KEylDS
KEylD следует за '/' например для: pub   1024D/D12B77CE - KEylD 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 KEylD                     # Показать отпечаток ключа
# gpg --edit-key KEylD                        # _едактировать ключ (например подпись
или добавить/удалить email)
```

VPN через SSH

С версии 4.3, *OpenSSH* поддерживает устройства *tun/tap*, позволяющие создавать шифрованный туннель, что может быть весьма полезно в случае удаленного администрирования. Это очень похоже на OpenVPN, основанный на *TLS*. Плюс протокола *SSH* в том, что для реализации не нужно устанавливать и настраивать дополнительный софт. Из минусов, низкая производительность на медленных линиях. Шифрованный туннель создается на основе одного *TCP* соединения, что весьма удобно, для быстрого поднятия простого *VPN*, на IP.

В конфигурационном файле *sshd_conf*, должны стоять следующие опции:

PermitRootLogin yes
PermitTunnel yes

Одно P2P (peer to peer — точка точка) соединение

Попробуем соединить два хоста *p2p_client* и *p2p_server*. Соединение инициирует *p2p_client* к *p2p_server*, при этом он должен обладать правами root. Конечные адреса туннеля 10.0.0.1 (сервер) и 10.0.0.2 (клиент), кроме того мы создаем устройство *tun5* (номер может быть любым). Вся процедура проста:

- Подключиться по SSH используя опцию -w

- Сконфигурировать IP адреса туннеля, *ssh* сервер — *ssh* клиент

Конфигурация SSH сервера

Подключение инициирует ssh клиент, все команды выполняются на сервере.

Сервер на Linux

```
cli>#  ssh -w5:5 root@p2p_server
srv>#  ifconfig tun5 10.0.1.1 netmask 255.255.255.252   # Выполняется в оболочке
сервера
```

Сервер на FreeBSD

```
cliуте#  ssh -w5:5 root@p2p_server
syrvук#  ifconfig tun5 10.0.1.1 10.0.1.2                  # Выполняется в системной
оболочке сервера
```

Конфигурирование SSH клиента

Команды выполняемые на *ssh* клиенте:

```
cli>#  ifconfig tun5 10.0.1.2 netmask 255.255.255.252   # SSH клиент на Linux
cli>#  ifconfig tun5 10.0.1.2 10.0.1.1                  # SSH клиент на FreeBSD
```

Теперь хосты соединены и могут обмениваться информацией, используя IP адреса туннеля.

Соединение двух сетей

Более полезная возможность *SSH*, соединение двух сетей, используя два шлюза. Предположим, есть две сети, сеть А с адресом 192.168.51.0/24 и сеть Б с адресом 192.168.16.0/24. Процедура идентична вышеописанной, за исключением того, что нужно будет добавить маршрут.
192.168.51.0/24 (сеть А)|шлюз А <-> шлюз Б|192.168.16.0/24 (сеть Б)

- Подключится через *SSH* с опцией -w.
- Настройка IP адреса SSH туннеля, делается единожды, на сервере и на клиенте.
- Добавить маршрут для обоих сетей.
- Если нужно, включить *NAT* на внутреннем интерфейсе шлюза.

Подключение из сети А к сети сети Б

Соединение начинается со шлюза А, команды выполняются на шлюзе Б.

шлюз Б на Linux

```
gateA>#  ssh -w5:5 root@gateB
gateB>#  ifconfig tun5 10.0.1.1 netmask 255.255.255.252 # Выполняется в оболочке шлюза
Б
gateB>#  route add -net 192.168.51.0 netmask 255.255.255.0 dev tun5
gateB>#  echo 1 > /proc/sys/net/ipv4/ip_forward        # Необходимо только если шлюз
не Default Gateway
gateB>#  iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

шлюз Б на FreeBSD

```
gateA>#  ssh -w5:5 root@gateB                          # Создаем устройство tun5
gateB>#  ifconfig tun5 10.0.1.1 10.0.1.2               # Выполняется на шлюзе Б
gateB>#  route add 192.168.51.0/24 10.0.1.2
gateB>#  sysctl net.inet.ip.forwarding=1               # Необходимо только если шлюз
не Default Gateway
gateB>#  natd -s -m -u -dynamic -n fxp0                # Смотрим NAT
gateA>#  sysctl net.inet.ip.fw.enable=1
```

Настройка шлюза A

Команды выполняемые на шлюзе A:

шлюз A на Linux

```
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
```

шлюз A на FreeBSD

```
gateA>#  ifconfig tun5 10.0.1.2 10.0.1.1
gateA>#  route add 192.168.16.0/24 10.0.1.2
gateA>#  sysctl net.inet.ip.forwarding=1
gateA>#  natd -s -m -u -dynamic -n fxp0
gateA>#  sysctl net.inet.ip.fw.enable=1
```

В итоге имеет две частные сети, прозрачно соединенные в VPN через SSH. Перенаправление IP и настройки NAT необходимы только если шлюзы не являются шлюзами по умолчанию (в этом случае клиент не будет знать, куда пересылать ответы).

Базы данных

PostgreSQL

Смена пароля root или пароля пользователя.

```
# psql -d template1 -U pgsql
> alter user pgsql with password 'pgsql_password';  # Используйте username вместо
"pgsql"
```

Создание пользователя и базы данных

Команды *createuser*, *dropuser*, *createdb* и *dropdb*, это эквиваленты *SQL* команд.

```
# createuser -U pgsql -P bob          # Создание пользователя bob,-P для ввода пароля.
# createdb -U pgsql -O bob bobdb      # Создать базу данных bobdb, владелец bob
# dropdb bobdb                        # Удалить базу дфнных bobdb
# dropuser bob                        # Удалить пользователя bob
```

Основной механизм авторизации в базе, настраивается в файле *pg_hba.conf*.

Разрешить удаленный доступ к базе данных

Файл *$PGSQL_DATA_D/postgresql.conf* определяет слушающие адреса. Обычно *listen_addresses = '*'* для Postgres 8.x.

Файл *$PGSQL_DATA_D/pg_hba.conf* назначает уровни доступа. Пример:

```
# TYPE  DATABASE    USER        IP-ADDRESS        IP-MASK           METHOD
host    bobdb       bob        212.117.81.42     255.255.255.255   password
host    all         all        0.0.0.0/0                           password
```

Резервное копирование и восстановление баз данных

Резервное копирование и восстановление делается пользователем *pgsql* или *postgres*, на пример:

```
# pg_dump --clean dbname > dbname_sql.dump
# psql dbname < dbname_sql.dump
```

Резервное копирование и восстановление всех баз (включая пользователей):

```
# pg_dumpall --clean > full.dump
# psql -f full.dump postgres
```

Базы данных MySQL

Смена пароля root или пароля пользователя.

Способ 1

```
# /etc/init.d/mysql stop
или# killall mysqld
# mysqld --skip-grant-tables
# mysqladmin -u root password 'newpasswd'
# /etc/init.d/mysql start
```

Способ 2

```
# mysql -u root mysql
mysql> UPDATE USER SET PASSWORD=PASSWORD("newpassword") where user='root'; #
Используйте имя "пользователя" вместо "root"
mysql> FLUSH PRIVILEGES;
mysql> quit
```

Создание пользователя создание базы данных.

```
# mysql -u root mysql
mysql> CREATE DATABASE bobdb;                        # Создать базу данных bobdb
mysql> GRANT ALL ON *.* TO 'bob'@'%' IDENTIFIED BY 'pwd'; # Используйте localhost
вместо % что-бы запретить доступ к базе данных извне
# % — разрешает сетевой доступ пользователя с любого IP адреса
mysql> DROP DATABASE bobdb;                          # Удалмть базу данных.
mysql> DROP USER bob;                                # Удалить пользователя bob.
mysql> DELETE FROM mysql.user WHERE user='bob and host='hostname'; # Альтернативы
mysql> FLUSH PRIVILEGES;
```

Разрешить удаленный доступ

Обычно удаленный доступ разрешен не ко всем базам данных. В файл */etc/my.cnf* прописан адрес для слушающего сокета, как правило достаточно раскомментировать строку *bind-address = *

```
# mysql -u root mysql
mysql> GRANT ALL ON bobdb.* TO bob@'xxx.xxx.xxx.xxx' IDENTIFIED BY 'PASSWORD';
mysql> REVOKE GRANT OPTION ON foo.* FROM bar@'xxx.xxx.xxx.xxx';
mysql> FLUSH PRIVILEGES;
```

Резевное копирование и восстановление баз данных mysql

Работа с одной базой данных:

```
# mysqldump -u root -psecret --add-drop-database dbname > dbname_sql.dump
# mysql -u root -psecret -D dbname < dbname_sql.dump
```

Работа со всеми базами данных:

```
# mysqldump -u root -psecret --add-drop-database --all-databases > full.dump
# mysql -u root -psecret < full.dump
```

В данном случае "*secret*", пароль пользователя root для mysql, после опции -p, пробел не ставится. Если опция -p будет использована без следеющего за ней пароля, он будет запрошен интерактивно.

Базы данных sqlite

SQLite небольшая, достаточно мощная, самодостаточная, неконфигурируемая база данных.

Дамп и восстановление

Вам может понадобится сделать дамп и восстановление базы SQLite. Например можно отредактировать файл дампа, поменять атрибуты или типы колонок а затем восстановить базу. Это проще чем ковыряться с SQL командами.

```
# sqlite database.db .dump > dump.sql             # Сделать резервную копию
# sqlite database.db < dump.sql                   # Восстановление дампа
```

Конвертирование 2.x базы в 3.x

```
sqlite database_v2.db .dump | sqlite3 database_v3.db
```

CVS

CVS — (*Concurrent Versions System, "Система Конкурирующих Версий"*) — программное решение из класса "систем управления версиями" (version control system). Содержит историю изменений определенного набора файлов, являющихся как правило исходниками какого-то програмного проекта. Позволяет вести совместную работу над проектом, группе людей.

Настройка CVS сервера

Подготовка CVS

Для начала нужно решить, где будет лежать основное хранилище и создать для него корневую директорию. Например */usr/local/cvs*:

```
# mkdir -p /usr/local/cvs
# setenv CVSROOT /usr/local/cvs      # Установить для переменной CVSROOT
расположение корня (local)
# cvs init                           # Создать все внутренние конфиги CVS
# cd /root
# cvs checkout CVSROOT               # Оформить рабочее дерево каталогов
# cd CVSROOT
edit config (fine as it is)
# cvs commit config
cat >> writers                       # Создать файл writers, содержащий пользователей
с правами на запись/изменение (опционально, файл readers)
colin
^D                                   # Используя Ctrl+D, выйти из режима
редактирования
# cvs add writers                    # Добавить разработчиков в репозитарий
# cvs edit checkoutlist
# cat >> checkoutlist
writers
^D                                   # Используя Ctrl+D, выйти из режима
редактирования
# cvs commit                         # Передать все сделанные изменения
```

Есть два способа разграничить права на чтение и запись репозитария. Включающий способ означает, что пользователи, имеющие права на чтение репозитария, явно указываются в файле *readers*. Исключающий способ означает, что права на запись репозитария, имеют только пользователи перечисленные в файле *writers*, остальные только на чтение.
Есть три способа получить доступ к репозитарию. Два первых не нуждаются в каком-либо дополнительном изменении конфигурации.

- Локальный доступ к файловой системе. Пользователь должен иметь достаточно прав для досткпа к *CVS*, ему достаточно авторизоваться в системе. Этот способ используется только для локального хранилища.
- Удаленный доступ через *SSH* туннель. Пользователь должен иметь *SSH* аккаунт и права на чтение/запись на *CVS* сервере. Это не требует запуска дополнительных процессов на *CVS* сервере, авторизация проходит через *SSH* соединение.
- Удаленный доступ с помощью *pserver* (порт по-умолчанию: 2401/tcp). Этот способ предпочтителен для большого кол-ва пользователей, авторизация проходит через *pserver*, пароли хранятся в отдельной базе данных, при этом нет надобности в локальных аккаунтах. Настройки данного способа приведены ниже.

Сетевые настройки с помощью inetd

CVS можно запустить локально, только если нет надобности в сетевом доступе. Для удаленного доступа, демон *inetd*, файл */etc/inetd.conf* (*/etc/xinetd.d/cvs* в *SuSE*), запускает *pserver*:

```
cvspserver stream tcp nowait cvs /usr/bin/cvs cvs --allow-root=/usr/local/cvs pserver
```

Неплохой идеей будет заблокировать порт *CVS* от прямого доступа из внешнего мира с помощью *firewall* и использовать *SSH* туннель.

Раздельная авторизация

Иногда может понадобиться, завести не локальных пользователей. Для этого просто добавте файл *passwd*, содержащий имена пользователей и пароли в зашифрованном виде, в директорию *CVSROOT*. Сделать это можно, например с помощью утилиты веб сервера Apache, *htpasswd*.

> Файл *passwd* можно редактировать напрямую, в директории *CVSROOT*. Для получения справки, используйте, *htpasswd --help*

```
# htpasswd -cb passwd user1 password1  # -c, создать файл паролей
# htpasswd -b passwd user2 password2
```

Теперь допишите *:cvs*, в конец каждой строки файла, что-бы *CVS* менял пользователя, от имени которого он запускается. Примерно так:

```
# cat passwd
user1:xsFjhU22u8Fuo:cvs
user2:vnefJOsnnvToM:cvs
```

Проверка работоспособности CVS

Попробуйте зайти с именем существующего пользователя:

```
# cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs login
Logging in to :pserver:colin@192.168.50.254:2401/usr/local/cvs
CVS password:
```

Переменная CVSROOT

Это переменная окружения, использующаяся для определения места хранения репозитария. Для локального использования, она может быть установлена в директории репозитария. Для сетевого доступа, должен быть определен транспортный протокол. Установите переменную *CVSROOT*, командой, *setenv CVSROOT* строка (для оболочек *csh* и *tcsh*), или export *CVSROOT=string* (для оболочек *sh* или *bash*).

```
# setenv CVSROOT :pserver:@:/cvsdirectory
# setenv CVSROOT /usr/local/cvs                                  # Только для локального
использования
# setenv CVSROOT :local:/usr/local/cvs                           # То-же, что и выше
# setenv CVSROOT :ext:user@cvsserver:/usr/local/cvs              # Прямой доступ по
протоколу SSH
# setenv CVS_RSH ssh                                             # Для ext доступа
# setenv CVSROOT :pserver:user@cvsserver.254:/usr/local/cvs      # Для сети, через
pserver
```

После авторизации в системе, можно импортировать новый проект:
Перейдите в корневую директорию проекта

```
cvs import
cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs import MyProject MyCompany START
```

Где, *MyProject*, имя вашего нового проекта в репозитарии (потом используется для проверки). *CVS* импортирует текущее содержимое директории в проект.

Проверяем:

```
# cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs checkout MyProject
или# setenv CVSROOT :pserver:colin@192.168.50.254:/usr/local/cvs
# cvs checkout MyProject
```

SSH туннель для CVS

Нам понадобится 2 оболочки. Из первой мы подключимся к CVS серверу по протоколу SSH и настроим форвардинг портов. Из второй оболочки будем подключаться к *CVS*, как будто он запущен на локальной машине.

Оболочка 1:

```
# ssh -L2401:localhost:2401 colin@cvs_server   # Прямое подключение к CVS серверу,
или:
# ssh -L2401:cvs_server:2401 colin@gateway     # Используйте шлюз
```

Оболочка 2:

```
# setenv CVSROOT :pserver:colin@localhost:/usr/local/cvs
# cvs login
Logging in to :pserver:colin@localhost:2401/usr/local/cvs
CVS password:
# cvs checkout MyProject/src
```

Основные команды CVS

Import

Команда *Import*, используется для добавления целой директории, вызывается из директории, которую нужно импортировать. К примеру, директория */devel/* содержит файлы и поддиректории для импорта. Директория в *CVS*, будет "*myapp*".

```
# cvs import [options] directory-name vendor-tag release-tag
# cd /devel                          # Войти в директорию проекта для импорта
# cvs import myapp Company R1_0      # Release tag может быть любым словом
```

После добавления новой директории, например */devel/tools/*, ее так-же можно импортировать.

```
# cd /devel/tools
# cvs import myapp/tools Company R1_0
```

Checkout update add commit

```
# cvs co myapp/tools                 # Проверка только директории tools
# cvs co -r R1_1 myapp               # Проверит "myapp", релиз R1_1
# cvs -q -d update -P                # Типичное обновление CVS
# cvs update -A                      # Сбросить sticky tag
# cvs add newfile                    # Добавить файл
# cvs add -kb newfile                # Добавить бинарный файл
# cvs commit file1 file2             # Передать только 2 файла
# cvs commit -m "message"            # Передать все изменения, сделанные в сообщении
```

Создание патча

Создавать патч, предпочтительней, из рабочей директории проекта или из исходников.

```
# cd /devel/project
# diff -Naur olddir newdir > patchfile # Создать патч из директории или файла
# diff -Naur oldfile newfile > patchfile
```

Применение патча

Иногда может понадобится отделить уровни директорий для патча, в зависимости от того, как он был создан. В случае затруднений, можно просто посмотреть первую строку в файле патча и попробовать варианты:-p0, -p1 или -p2.

```
# cd /devel/project
# patch --dry-run -p0 < patchfile    # Проверить патч, не применяя его
# patch -p0 < patchfile
# patch -p1 < patchfile              # Отделить один уровень от пути
```

RSYNC

Программа *Rsync* (во FreeBSD есть в портах), используется для удаленного копирования (резервного копирования) или синхронизации файлов и каталогов, с минимальными затратами трафика. Может практически целиком заменить *cp* и *scp*, умеет кодировать данные, поддерживает сжатие и рекурсию, кроме того, прерванные передачи можно с легкостью перезапустить. На страницах руководства, все описано довольно подробно.Вот несколько примеров:

Копировать директорию с контентом:

```
# rsync -a /home/colin/ /backup/colin/
# rsync -a /var/ /var_bak/
# rsync -aR --delete-during /home/user/ /backup/      # Используется относительный
путь (см. ниже)
```

То-же что и выше, только по сети и с компрессией. По-умолчанию, Rsync использует для передачи протокол *SSH* в том числе и с ключами, если таковые имеются. Символ ":" используется как в *SCP*. Типичный пример удаленного копирования:

```
# 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/*. Данный пример типичен для резервного копирования:

```
# rsync -azR --exclude /tmp/ /home/user/ user@server:/backup/
```

Использовать 20022 порт для *SSH*:

```
# rsync -az -e 'ssh -p 20022' /home/colin/ user@server:/backup/colin/
```

Можно использовать демон *rsync* (с "::"), это гораздо быстрее, но трафик не шифруется. Местонахождение папки 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/ ruser@hostname::rmodule/backup/
# rsync -axSRz ruser@hostname::rmodule/backup/ /home/    # Копировать обратно
```

Некоторые важные опции:

- -a, --archive — режим архива; то-же что и -rlptgoD (без -H)
- -r, --recursive — обходить директории (рекурсия)
- -R, --relative — относительные пути
- -H, --hard-links — сохранять жесткие ссылки (*hardlink*)
- -S, --sparse — handle sparse files efficiently
- -x, --one-file-system — не пересекать границы файловой системы
- --exclude=PATTERN — исключить файлы заданного образца
- --delete-during — приемник удаляется ПРИ ПЕРЕДАЧЕ
- --delete-after — приемник удаляется ПОСЛЕ ПЕРЕДАЧИ

winrsync

Под Windows, *rsync* можно использовать через *cygwin* или отдельным приложением cwrsync. Очень удобно для автоматизации резервного копирования. Установите ОДИН из вариантов и добавьте путь в системные переменные *Windows: Control Panel -> System -> tab Advanced, button Environment Variables*. Отредактируйте переменную "*Path*", добавив полный путь до *rsync*, например так: *C:\Program Files\cwRsync\bin* или *C:\cygwin\bin*. Это позволит использовать *rsync* и *ssh* из командной строки Windows.

Авторизация по ключу

Rsync автоматически туннелируется через *SSH* протокол, а тот использует *SSH* авторизацию на сервере. Автоматическое резервное копирование используется для минимизации участия пользователя в этом процессе, как раз для этого и нужна авторизация по публичному ключу, что-бы не запрашивать у пользователя ввод пароля.
Все команды выполняются в командной оболочке Windows (*Start -> Run -> cmd*). Создайте и загрузите ключи, как описано в материале SSH, SCP, "*user*" и "*server*" установите соответствующие.

```
# ssh-keygen -t dsa -N ''                    # Создаем ключи
# rsync user@server:.ssh/authorized_keys2    # Копируем файл ssh/authorized_keys2 на
локальную машину
# cat id_dsa.pub >> authorized_keys2         # Или используйте редактор, что-бы
добавить ключ
# rsync authorized_keys2 user@server:.ssh/   # Копируем файл обратно на сервер
# del authorized_keys2                       # Удаляем локальную копию
```

Теперь проверяем (одной строкой):

```
rsync -rv "/cygdrive/c/Documents and Settings/%USERNAME%/My Documents/"
'user@server:My\ Documents/'
```

Автоматизация резервного копирования

Планировка и выполнение задания резервного копирования, можно вполне возложить на какой-нибудь планировщик или пакетные файлы (Programs -> Accessories -> System Tools -> Scheduled Tasks). Например, можно создать такой файл, заменив "user@server" на свои:

```
@ECHO OFF
REM rsync the directory My Documents
SETLOCAL
SET CWRSYNCHOME=C:\PROGRAM FILES\CWRSYNC
SET CYGWIN=nontsec
SET CWOLDPATH=%PATH%
REM uncomment the next line when using cygwin
SET PATH=%CWRSYNCHOME%\BIN;%PATH%
echo Press Control-C to abort
rsync -av "/cygdrive/c/Documents and Settings/%USERNAME%/My Documents/" \
'user@server:My\ Documents/'
pause
```

ssh авторизация по ключам, безопасное копирование scp

Public key — Публичный ключ

При удаленном администрировании серверов, очень удобно использовать авторизацию по паре ключей, что-бы не вводить пароль при каждом соединении с удаленной машиной. Для соединения с удаленным хостом без ввода пароля, можно использовать так называемый *Public key* (*публичный ключ*). Нужно добавить запись о публичном ключе в файл *~/.ssh/authorized_keys* на удаленном SSH сервере. В данном случае для подключения используются ключи сгенерированные на стороне клиента. В *cygwin* создать директорию */home* и *.csh* в ней, можно с помощью *# mkdir -p /home/USER/.ssh*

- Используя *ssh-keygen*, генерируем пару ключей. *~/.ssh/id_dsa*-приватный ключ, *~/.ssh/id_dsa.pub*-публичный ключ.
- Копируем только публичный ключ на сервер и добавляем его в файл *~/.ssh/authorized_keys2*, *~/* -домашняя директория.

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

Используя Windows клиент с ssh.com

Коммерческую версию клиента, можно скачать с официального сайта:ssh.com. Ключи сгенерированные с помощью ssh.com клиента нужно конвертировать в формат OpenSSH сервера, это делается с помощью *ssh-keygen*.

- Создать пару ключей с помощью ssh.com клиента: *Settings — User Authentication — Generate New....*
- Тип ключа *DSA*; Длина ключа *2048*.
- Скопировать публичный ключ на сервер в папку *~/.ssh*.
- Ключи находятся в *C:\Documents and Settings\%USERNAME%\Application Data\SSH\UserKeys*.
- Используя *ssh-keygen* на сервере, конвертируем ключи:

```
# cd ~/.ssh
# ssh-keygen -i -f keyfilename.pub >> authorized_keys2
```

Использовать можно как *DSA*, так и *RSA* алгоритм. Имейте в виду, сгенерированные нами ключи не защищены паролем, а это определенный минус для безопасности.

Использование putty для Windows

Putty, отличный бесплатный клиент под Windows.

- Создать пару ключей с помощью утилиты *puttygen* (идет в комплекте с полным дистрибутивом)
- Сохранить публичный и приватный ключи, например в *C:\Documents and Settings\%USERNAME%\.ssh*.
- Копировать публичный ключ на сервер в папку *~/.ssh*:

```
# scp .ssh/puttykey.pub root@192.168.51.254:.ssh/
```

- Используя *ssh-keygen* на сервере, конвертировать ключи в формат OpenSSH:

```
# cd ~/.ssh
# ssh-keygen -i -f puttykey.pub >> authorized_keys2
```

- В *PuTTy: Connection — SSH — Auth*, прописать путь к приватному ключу.

Fingerprinf — Проверка отпечатка

При первом подключении, *ssh* проверит, если хост с данным отпечатком неизвестен, будет запрошено подтверждение *yes/no*, если ответ будет *yes*, хост будет добавлен в файл *~/.ssh/known_hosts* и при следующих подключениях будет сразу запрошен пароль для подключения, если ответ будет *no*, соединение будет разорвано.

```
# ssh test_host
The authenticity of host 'test_host (192.168.16.54)' can't be established.
DSA key fingerprint is 14:4a:aa:d9:73:25:46:6d:0a:48:35:c7:f4:16:d4:ee.
Are you sure you want to continue connecting (yes/no)? yes
```

Получить отпечаток можно командой *ssh-keygen -l* на сервере:

```
# ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub      # Для RSA ключа
2048 61:33:be:9b:ae:6c:36:31:fd:83:98:b7:99:2d:9f:cd /etc/ssh/ssh_host_rsa_key.pub
# ssh-keygen -l -f /etc/ssh/ssh_host_dsa_key.pub      # Для DSA ключа (по-умолчанию)
2048 14:4a:aa:d9:73:25:46:6d:0a:48:35:c7:f4:16:d4:ee /etc/ssh/ssh_host_dsa_key.pub
```

SCP — Безопасное копирование

Пара простых команд:

```
# scp file.txt remote_host:/tmp
# scp joe@remote_host:/www/*.html /www/tmp
# scp -r joe@remote_host:/www /www/tmp
```

Еще пара примеров использования SCP

```
# scp -P 2525 ./file_to_copy root@remote_host:/tmp/copied_file
```

В данном случае, у нас удаленный сервер SSH, сидит на нестандартном порту, поэтому с помощью опции -P, указываем на какой порт подключаемся. Не путайте с ключем -p у программы *ssh*, для указания порта, для *scp*, опция -p, означает "Сохранить время модификации, время последнего доступа, и режимы, оригинального файла". В случае, если у пользователя, под которым вы пытаетесь подключится для копирования, установлен пароль (что собственно и

должно быть), *SCP* попросит его ввести, выглядит это примерно так.

```
# scp -P 2222 ./file_to_copy user@remote_host:~/copied_file
# Password:                                                    вводим пароль, жмем
Ентер
# copied_file
```

Подключить удаленную папку можно с помощью *Konqueror* или *Midnight Commander*, введя путь в виде user@gate. Кроме того можно воспользоваться утилитой sshfs, это клиент для файловых систем на базе SCP. See fuse sshfs.

Создание защищенного SSH туннеля

SSH туннелирование позволяет делать форвардинг портов (перенаправление) через *SSH* туннель, обеспечивая прохождение трафика через заблокированые порты (работает только с протоколом *TCP*).
SSH Туннель создается из слушающего сокета на определенном порту *localhost*. Затем все принятые на локальном хосте/порту соединения перенаправляются через *SSH* на удаленный хост/порт.

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

Прямой форвардинг на шлюз

Предположим, нам нужно получить доступ к *CVS* (порт 2401) и *http* (порт 80), запущенных на удаленном хосте. Ниже вы видите простой пример реализации, 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 удаленного рабочего стола

Имеем Windows SMB сервер за шлюзом, и отсутствие *ssh*. Необходимо получить доступ к расшаренным *SMB* папкам и удаленному рабочему столу.

```
# 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.

- With putty use Source port=10.1.1.1:139. It is possible to create multiple loop devices and tunnel. On Windows 2000, only putty worked for me. On Windows Vista also forward the

port 445 in addition to the port 139. Also on Vista the patch KB942624 prevents the port 445 to be forwarded, so I had to uninstall this path in Vista.
- With the ssh.com client, disable "Allow local connections only". Since ssh.com will bind to all addresses, only a single share can be connected.

Теперь 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.

Перезагружаемся. Теперь можно подключиться к расшаренным ресурсам по адресу \\10.1.1.1 и к удаленному рабочему столу по адресу 10.1.1.1:3388.

Отладка

Если что-то не работает:

- Порты проброшены правильно, *netstat -an*? ищите 0.0.0.0:139 или 10.1.1.1:139
- Попробуйте подключиться через telnet по адресу 10.1.1.1 139
- Должна быть включена опция "Local ports accept connections from other hosts".
- Сервис "File and Printer Sharing for Microsoft Networks" отключен на loopback интерфейсе?

Подключение двух клиентов, сидящих за NAT

Есть две машины, находящиеся за NAT шлюзом, клиенту *cliadmin* нужно подключится к клиенту *cliuser*, оба имеют доступ к Linux-шлюзу по *ssh*. Поскольку будут использованы порты, выше 1024, root доступ не понадобится.
На шлюзе мы используем порт 2022.
На клиенте *cliuser*:

```
# ssh -R 2022:localhost:22 user@gate            # Форвардинг порта клиента 22, на порт 2022, шлюза
```

На клиенте *cliadmin*:

```
# ssh -L 3022:localhost:2022 admin@gate         # Форвардинг порта клиента 3022, на порт 2022 шлюза
```

Теперь администратор может напрямую подключится к *cliuser*:

```
# ssh -p 3022 admin@localhost                   # local:3022 -> gate:2022 -> client:22
```

Подключение к рабочему столу, расположенному за NAT

Предположим нужно получить доступ к Windows клиенту с *VNC* слушающем на 5900 порту.
От клмента *cliwin* на шлюз:

```
# ssh -R 15900:localhost:5900 user@gate
```

От клиента *cliadmin*:

```
# ssh -L 5900:localhost:15900 admin@gate
```

Теперь админ может напрямую подключится к клиентскому *VNC*:

```
# vncconnect -display :0 localhost
```

Отладка multi-hop ssh tunnel

Предположим, вы не можете получить прямой доступ к *ssh*, только через промежуточные хосты (например из-за проблем с маршрутизацией), но получить соединение клиент-сервер вам 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 в одной оболочке:
клиент -> хост1 -> хост2 -> сервер and dig tunnel 5678

```
Клиент>$ ssh -L5678:localhost:5678 host1        # 5678 произвольный порт для туннеля
хост_1>$ ssh -L5678:localhost:5678 host2        # звено 5678 с хохта1 на хост2
хост_2>$ ssh -L5678:localhost:22 server         # и тцннель на порт 22 сервера
```

Использование туннеля через другой шелл

клиент -> сервер использующий туннель 5678

```
Клиент>$ ssh -L5678:localhost:5678 host1        # 5678 произвольный порт для туннеля
хост_1>$ ssh -L5678:localhost:5678 host2        # звено 5678 с хохта1 на хост2
хост_2>$ ssh -L5678:localhost:22 server         # и тцннель на порт 22 сервера
```

SUDO

Sudo — стандартное средства наделить пользователя некоторыми административными правами доступа, без выдачи ему пароля на root. Sudo весьма полезна, когда на удаленном сервере работает много пользователей. Вызывается она просто:# sudo /etc/init.d/dhcpd restart # Запустить стартовый скрипт от имени root # sudo -u sysadmin whoami # Запустить команду от имени другого пользователя

Конфигурация Sudo

Программа *sudo* конфигурируется в файле */etc/sudoer*s и зачастую только с помощью специального редактора *visudo*. Базовый синтаксис (разделитель в строке, знак ","):

```
user hosts = (runas) commands          # B /etc/sudoers
```

- users один или более пользователей или группа (например *%wheel*) для расширения прав доступа
- hosts списак 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

Кроме того существуют псевдонимы *User_Alias, Host_Alias, Runas_Alias* и *Cmnd_Alias*. Полезно, если конфиг слишком большой. Пример файла *sudoers*:

```
# cat /etc/sudoers
# Псевдонимы хоста, подсети или имена хостовHost_Alias   DMZ     = 212.118.81.40/28
Host_Alias   DESKTOP = work1, work2
# Псевдонимы пользователя, список пользователей имеющих некоторые права
доступаUser_Alias   ADMINS  = colin, luca, admin
User_Alias   DEVEL   = joe, jack, julia
Runas_Alias  DBA     = oracle,pgsql
# Псевдонимы команд, список полных путей до командCmnd_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 # Not root pwd!
Cmnd_Alias   DEBUG   = /usr/sbin/tcpdump,/usr/bin/wireshark,/usr/bin/nmap
# Актуальные права доступаroot,ADMINS  ALL     = (ALL) NOPASSWD: ALL    # ADMINS
может что-то делать без пароля
DEVEL        DESKTOP = (ALL) NOPASSWD: ALL    # _азработчики имеют полные права
доступа на рабочих станциях
DEVEL        DMZ     = (ALL) NOPASSWD: DEBUG  # _азработчики могут отлаживать DMZ
сервера.
# Пользователь sysadmin может использовать некоторые командыsysadmin     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 станцияхALL          DESKTOP
= NOPASSWD: /sbin/mount /cdrom,/sbin/umount /cdrom
```

SVN

Настройка SVN сервера

Настроить сервер — SVN репозитарий, довольно просто, в данном примере, директория, */home/svn/*, должна существовать:

```
# svnadmin create --fs-type fsfs /home/svn/project1
```

Подключиться к репозитарию можно следующими способами:

- file:// — Прямой доступ через файловую систему с помощью *SVN* клиента. В локальной файловой системе должны быть корректно настроены привилегии.
- svn:// или svn+ssh:// — Удаленный доступ к *SVN* серверу (так-же по протоколу *SSH*). Требуются права в локальной файловой системе, порт по-умолчанию: 2690/tcp.
- http:// Удаленный доступ через *webdav*, используя *apache*. Для данного метода не требуется наличие локальных пользователей.

Импорт и проверка существующего проекта через локальную файловую систему. В рабочую директорию заходить не обязательно, можно просто указать полный путь:

```
# svn import /project1/ file:///home/svn/project1/trunk -m 'Initial import'
# svn checkout file:///home/svn/project1
```

Удаленный доступ по протоколу SSH

Удаленный доступ по протоколу *SSH* не требует каких-то доаолнительных настроек, просто замените *file://* на *svn+ssh/hostname*. Например:

```
# svn checkout svn+ssh://hostname/home/svn/project1
```

Как и в случае локального доступа, пользователь должен иметь аккаунт, для доступа по протоколу *SSH* на сервер, и корректно настроенные права на чтение/запись. Данный метод может быть пригоден для небольших групп пользователей, пользователи входящие в группу, являются владельцами хранилища, например:

```
# groupadd subversion
# groupmod -A user1 subversion
# chown -R root:subversion /home/svn
# chmod -R 770 /home/svn
```

Удаленный доступ через HTTP (apache)

Удаленный доступ черз *HTTP* (*HTTPS*), подходящее решение для удаленных групп пользователей. Этот метод использует авторизацию веб сервера *Apache*(не локальные аккаунты). Вот типичная конфигурация Apache:

```
LoadModule dav_module         modules/mod_dav.so
LoadModule dav_svn_module     modules/mod_dav_svn.so
LoadModule authz_svn_module   modules/mod_authz_svn.so    # Только для контроля
доступа


DAV svn
# URL "/svn/foo" будет отображен на путь репозитария /home/svn/foo
SVNParentPath /home/svn
AuthType Basic
AuthName "Subversion repository"
AuthzSVNAccessFile /etc/apache2/svn.acl
AuthUserFile /etc/apache2/svn-passwd
Require valid-user
```

Сервер *Apache* должен иметь полный доступ к репозитарию:

```
# chown -R www:www /home/svn
```

Создание пользователя с помощью *htpasswd*:

```
# htpasswd -c /etc/svn-passwd user1  # -c Создать файл
```

Контроль доступа svn.acl

```
# По-умолчанию доступ на чтение. "* =" по-умолчанию не будет иметь доступа
[/]
* = r
[groups]
project1-developers = joe, jack, jane
# Дать права на запись разработчикам[project1:]
@project1-developers = rw
```

Некоторые команды для управления SVN репозитарием

Смотрите так-же Subversion Quick Reference Card. Tortoise SVN, неплохой Windows интерфейс.

Импорт

Импортировать в репозитарий новый проект, содержащий директории и файлы, можно с помощью команды *import*. Эта-же команда используется и для добавления директории с ее содержимым в уже существующий проект.

```
# svn help import                                  # Помощь по команде
# Добавить новую директорию и ее содержимое в директорию src, проекта project1.
# svn import /project1/newdir http://host.url/svn/project1/trunk/src -m 'add newdir'
```

Типичные команды SVN

```
# svn co http://host.url/svn/project1/trunk      # Оформить заказ на последнюю версию
# Тэги и ветки создаются с помощью копирования
# svn mkdir http://host.url/svn/project1/tags/   # Создать директорию tags
# svn copy -m "Tag rc1 rel." http://host.url/svn/project1/trunk \
http://host.url/svn/project1/tags/1.0rc1
# svn status [--verbose]                         # Проверить состояние файлов в
рабочей директории
# svn add src/file.h src/file.cpp                # Добавить два файла
# svn commit -m 'Added new class file'           # Передать изменения сообщением
# svn ls http://host.url/svn/project1/tags/      # Список всех тэгов
# svn move foo.c bar.c                           # Переместить (переименовать) файлы
# svn delete some_old_file                       # Удалить файлы
```

Дисковые квоты

Дисковые квоты позволяют ограничить дисковое пространство или количество файлов использкемых пользователем (или членом группы). Квоты распределяются на уровне файловой системы и поддерживаются ядром.

Дисковые квоты в Linux

Пакет *quota tools*, как правило нуждается в установке, он содержит утилиты командной строки. Сначала нужно активировать дисковые квоты в файле *fstab* и перемонтировать раздел. Если раздел заблокирован открытыми файлами, нужно перезагрузить систему и добавить опцию монтирования *usrquota* в файл *fstab*.

```
/dev/sda2     /home    reiserfs     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.
```

Активировать квоты с помощью скрипта (например: */etc/init.d/quotad в SuSE*) или quotaon:

```
quotaon -vu /home
```

Проверить активацию:

```
quota -v
```

Дисковые квоты FreeBSD

Инстркменты дисковых квот в операционной системе FreeBSD являются частью базовой системы, однако в ядро должно быть включено *option quota*. Если это не так, добавте необходимую опцию и перекомпилируйте ядро.

```
options QUOTA
```

Как и в Linux, нужно добавить в *fstab* соответствующую опцию *userquota*:

```
/dev/ad0s1d    /home    ufs     rw,noatime,userquota    2  2
# mount /home                          # Перемонтировать раздел
```

Включение квот в файле в */etc/rc.conf*.

```
# grep quotas /etc/rc.conf
enable_quotas="YES"                    # Активировать дисковые квоты при запуске
системы (или NO).
check_quotas="YES"                     # Проверять квоты при старте (или NO).
# /etc/rc.d/quota start
```

Ограничения квот

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*. Например:

```
# edquota -u colin
```

Linux

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

FreeBSD

```
Quotas for user colin:
/home: kbytes in use: 504184, limits (soft = 700000, hard = 800000)
inodes in use: 1792, limits (soft = 0, hard = 0)
```

Изменение дисковых квот для нескольких пользователей

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

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

Проверка квот

Пользователи могут проверить свои квоты командой *quota* (файл *quota.user* должен быть доступен для чтения). Пользователь root может проверять любые квоты.

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

Shell скрипты

Bourne shell (*/bin/sh*) присутствует во всех Unix системах, соответственно скрипты, написанные на этом языке, будут работать на любой Unix-машине. Для прочтения: *man 1 sh*. Незаменимая вещь при настройке и обслуживании сервера.

Основые понятия о shell скриптах

Переменные и аргументы командной строки

Присваивание значений переменным производится следующим образом: *variable=value*, получить присвоенное значение можно по ссылке *$variable*.

```
MESSAGE="Hello World"                       # Присвоить переменной строку "Hello
World"
PI=3.1415                                   # Присвоить цифровое значение
N=8
TWON=`expr $N * 2`                          # Присвоить арифметическое выражение
(только целые числа)
TWON=$(($N * 2))                            # Другой вариант
TWOPI=`echo "$PI * 2" | bc -l`              # Использование bc для операций с
плавающей точкой
ZERO=`echo "c($PI/4)-sqrt(2)/2" | bc -l`
```

Аргументы командной строки:

```
$0, $1, $2, ...                             # $0 — сама команда(название скрипта)
$#                                          # Кол-во аргументов командной строки
$*                                          # Все аргументы (аналог $@)
```

Специальные переменные

```
$$                                          # ID текущего процесса
$?                                          # Код возврата последней выполненной
команды
command
if [ $? != 0 ]; then
echo "command failed"
fi
mypath=`pwd`
mypath=${mypath}/file.txt
echo ${mypath##*/}                          # Вывести только имя файла
echo ${mypath%%.*}                          # Полный путь без расширения
var2=${var:=string}                         # Если var назначена, использовать ее
значение, иначе string
# значение string будет присвоено и var и var2.
```

Конструкции

```
for file in `ls`
do
echo $file
done
count=0
while [ $count -lt 5 ]; do
echo $count
sleep 1
count=$(($count + 1))
done
myfunction() {
find . -type f -name "*.$1" -print       # $1 -первый аргумент функции
}
myfunction "txt"
```

Простенький скрипт генерирующий файл

```
MYHOME=/home/colin
cat > testhome.sh << _EOF   # Все до символа _EOF, записывается в файл testhome.sh
if [ -d "$MYHOME" ] ; then
echo $MYHOME exists
else
echo $MYHOME does not exist
fi
_EOF
sh testhome.sh
```

Еще один пример скрипта на bourne shell

В данном примере, скрипт создает *PDF* буклет, из *XHTML* документа:

```
#!/bin/sh
# Данный скрипт создает книгу в формате PDF, для печати на принтереif [ $# -ne 1 ];
then                          # Проверить аргумент
echo 1>&2 "Usage: $0 HtmlFile"
exit 1                                    # Выход с кодом больше нуля в случае
ошибки
fi
file=$1                                     # Присвоить имя файла из агрумента
fname=${file%.*}                            # Взять только имя файла
fext=${file#*.}                             # Взять только расширение
prince $file -o $fname.pdf
pdftops -paper A4 -noshrink $fname.pdf $fname.ps # Создать postscript буклет
cat $fname.ps |psbook|psnup -Pa4 -2 |pstops -b "2:0,1U(21cm,29.7cm)" > $fname.book.ps
ps2pdf13 -sPAPERSIZE=a4 -sAutoRotatePages=None $fname.book.ps $fname.book.pdf
# В Windows используйте #a4 и #None !exit 0                                    #
Выход с кодом успешного завершения
```

awk

Awk — это весьма мощьный и полезный язык для обработки текстовой информации. Кучу примеров, без труда можно найти в сети, здесь приведены лишь несколько простых:

```
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
```

sed

Sed — это неинтерактивный строчный редактор, принимает текст с устройства stdin или из текстового файла, выполняет некоторые со строками и выводит в stdout или в файл. Часто применяется в конвейерной обработке данных совместно с другими командами.

```
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
```

regex — 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
```

Некоторые полезные команды

Следующие команды могут пригодиться для использования как в скриптах, так и просто из командной строки:

```
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 $INTERFACE | sed '/.*inet addr:/!d;s///;s/ .*//')   # Linux
IP=$(ifconfig $INTERFACE | sed '/.*inet /!d;s///;s/ .*//')        # FreeBSD
if [ `diff file1 file2 | wc -l` != 0 ]; then [...] fi       # Файл изменен?
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 :). Убьет вашу машину))
tail +2 file > file2                           # Удалить первую строку из файла
```

Трюк, что-бы разом изменить расширение пачки файлов, например с *.cxx на *.cpp. (сначала попробуйте без | sh в конце строки). Тоже самое можно сделать с помощью команды rename, если она присутствует, или с помошью встроенных средст оболочки.

```
# ls *.cxx | awk -F. '{print "mv "$0" "$1".cpp"}' | sh
# ls *.c | sed "s/.*/cp & &.$(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
```

Полезные команды

less

Команда less используется для просмотра текстового документа, выводимого в stdout. Присутствует в большинстве дистрибутивов.

```
# less unixtoolbox.xhtml
```

Основные команды управления (^N — Ctrl + N):

- h H — Помощь
- f ^F ^V SPACE — Листать вВперед на один экран (или N строк).
- b ^B ESC-v — Листать азад на один экран (или N строк).
- F — Листать постоянно; Эквивалент "tail -f".
- /pattern — Искать вперед N-е совпадение строк
- ?pattern — Искать вперед N-е совпадение строк
- n — Повторить предыдущюю операцию поиска
- N — Поторить операцию поиска в обоих направлениях
- q — Выйти

vi

Текстовый редактор Vi присутствует в любой Unix подобной системе, поэтому будет нелишним, знать его основные команды. В *VI*, есть два режима работы, командный и режим вставки. В коммандный режим можно перейти, нажав [ESC], в режим вставки с помощью i. Для получения помощи, используйте : help.
Редактолры *nano* и *pico*, не менее распространены, и попроще в использовании (прим. автора), не согласен (прим переводчика).

Выход

- :w newfilename — Созранить файл с именем *newfilename*
- :wq или :x — Сохранить и выйти
- :q! — Выйти без сохранения

Поиск и перемещение

- /string — Поиск вниз, строки "string"
- ?string — Поиск вверх, строки "string"
- n — Поиск следующего совпадения

- N — Поиск предыдущего совпадения
- { — Перейти на параграф назад
- } — Перейти на параграф вперед
- 1G — Перейти на первую строку файла
- nG — Перети на строку №
- G — Перейти на последнюю строку
- :%s/OLD/NEW/g — Найти и заменить каждое совпадение

Удаление текста

- dd — Удалить текущую строку целиком
- D — Удалить текст, оставив пустую строку
- dw — Удалить слово
- x — Удалить букву
- u — Откатить последнюю операцию
- U — Откатить все изменения в текущей строке

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).
Пример:

```
# mail c@cb.vu
Subject: Your text is full of typos
"For a moment, nothing happened. Then, after a second or so,
nothing continued to happen."
.
EOT
#
```

Можно использовать конвейер (*pipe*), символ "|":

```
# echo "This is the mail body" | mail c@cb.vu
```

Так-же, это простой способ проверить почтовый сервер.

Программа tar, для создания архивов

Команда *tar*, предназначена для архивирования файлов или директорий. Имейте в виду, сам по себе *tar*, это не сжатый архив, сжатые архивы имеют расширения *.tgz* или *.tar.gz* (gzip) или *.tbz* (bzip2). Не используйте абсолютный путь при создании архива, возможно вы захотите распаковать его в другое место. Примеры использования:

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
```

Включить в архив одну директорию (или несколько) из дерева, но сохранить относительные пути. Например, необходимо создать архив, содержащий директории */usr/local/etc* и */usr/local/www*, директория local/ должна быть началом дерева.

```
# tar -C /usr -czf local.tgz local/etc local/www
# tar -C /usr -xzf local.tgz    # _аспаковать архив директорию local в дерево /usr
# cd /usr; tar -xzf local.tgz   # То-же, что выше
```

Распаковка архива tar

```
# tar -tzf home.tgz             # Просмотр содержимого архива без его распаковки
(листинг)
# tar -xf home.tar              # _аспаковать 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/colin/file.txt    # _аспаковать один файл
```

Дополнительно

```
# 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/
```

dd

Программа *dd* (*disk dump*), используется для копирования (конвертирования) дисков, разделов, и прочих операций копирования. Типичный пример:

```
# dd if= of= bs= conv=
```

Важные опции для *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 eedcoba@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.
```

Лечение

Программа *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 \ # Отправить на
удаленный хост
root@fry 'dd of=hda1.gz bs=1k'
# 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)
# kill -USR1 PID                                       # Посмотреть текущее состояние
dd (Linux)
# kill -INFO PID                                       # Посмотреть текущее состояние
dd (FreeBSD)
```

Трюки с MBR (master boot record)

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

Screen — оконный менеджер виртуальных терминалов

Screen имеет две основные функциональности:

- Запуск нескольких сессий терминала, в одном окне.
- Запуск программ отдельно от терминала в фоновом режиме. Терминал может быть отключен и переподключен позже.

Запустить screen можно командой:

```
# screen
```

Со *screen* сессией удобно использовать программы с длинным листингом (например *Top*).

```
# top
```

Теперь отсоединим screen от физического терминала с помощью Ctrl-a Ctrl-d.
Переподключиться можно:

```
# screen -rd
```

В частности это означает: если сессия запущена, переподключаться, если нужно переподключиться удаленно, сначала выйти из систему, если не запущена, запустить и уведомить пользователя. Или:

```
# screen -x
```

Подключиться к screen в многоэкранном режиме. Консоль могут использовать несколько пользователей, полезно для совместной работы.

Основные команды screen

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

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

Программа find

Важные опции:

- -x (в BSD) -xdev (в Linux) — 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 но без учета регистра

- -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 файлы с правами 0444
# find . -type d ! -perm -111         # Найти директории с правами 0111
# find /home/user/ -cmin 10 -print    # Файлы созданные или модифицированные за
последние 10 минут.
# find . -name '*.[ch]' | xargs grep -E 'expr' # Найти 'expr' в текущей директории.
# find / -name "*.core" | xargs rm    # Найти и удалить аварийные дампы(так-же можно
искать core.*).
# find / -name "*.core" -print -exec rm {} \;  # Другой сиснтаксис
# Найти все графические файлы и создать архив, iname -регистронезависимо. -r —
добавить
# find . \(-iname "*.png" -o -iname "*.jpg" \) -print -exec tar -rf images.tar {} \;
# find . -type f -name "*.txt" ! -name README.txt -print  # Исключая файлы README.txt
# find /var/ -size +10M -exec ls -lh {} \;      # Найти файлы больше 10 MB
# find /var/ -size +10M -ls            # То-же, что и выше
# find . -size +10M -size -50M -print
# find /usr/ports/ -name work -type d -print -exec rm -rf {} \;  # Очистить порты
# 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. Мини урок по команде find.

```
# 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 '{}'
```

Разное

```
# which command                       # Показывает полный путь к файлу команды
# time command                        # Показывает время выполнения команды
# time cat                            # Использовать команду time как секундомер. Ctrl-
c для остановки
# set | grep $USER                    # Просмотр текущего окружения
# cal                                 # Показать календарь на текущий месяц
# date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
# date 10022155                       # Установить дату и время
# whatis grep                         # Показать короткую справку по команде
# whereis java                        # Найти путь и стандартную директорию для
"слова"
# setenv varname value                # Установить переменную окружения varname в
значение value (csh/tcsh)
# export varname="value"              # Установить переменную окружения varname в
значение value (sh/ksh/bash)
# pwd                                 # Печать текущей директории
# mkdir -p /path/to/dir               # Создать директорию, включая родительскую, не
выдавать ошибку если такая существует.
# mkdir -p project/{bin,src,obj,doc/{html,man,pdf},debug/some/more/dirs}
# rmdir /path/to/dir                  # Удалить директорию.
# rm -rf /path/to/dir                 # Удалить директорию с содержимым
(принудительно).
# cp -la /dir1 /dir2                  # Вместо копирования отобразить одну директорию
в другую с помощью жесткой ссылки
# cp -lpR /dir1 /dir2                 # То-же во FreeBSD
# cp unixtoolbox.xhtml{,.bak}         # Быстрый вариант скопировать файл с новым
расширением
```

```
# mv /dir1 /dir2                    # Переименовать директорию
# ls -1                             # Лстинг файлов, по одному в строке
# history | tail -50                # Показать последние 50 использовавшихся команд
```

Проверить хэш файла с помощью OpenSSL. Неплохая альтернатива программам md5sum и sha1sum.

```
# openssl md5 file.tar.gz           # Сгенерировать md5 файла
# openssl sha1 file.tar.gz          # Сгенерировать sha1 файла
# openssl rmd160 file.tar.gz        # Сгенерировать RIPEMD-160 файла
```

Оболочки

Во многих дистрибутивах Linux, в качестве системной оболочки, используется *bash*, в BSD семействе, в основном *tcsh*, *bourne shell* используется только для скриптов.

Фильтры, весьма полезная штука при работе в системной оболочке, могут работать через конвейер "|":

- grep — Совпадение с образцом
- sed — Найти и заменить строки или символы
- cut — Печать определенной колонки из совпадения
- sort — Цифровая или алфавитная сортировка
- uniq — Удалить из вывода(файла) дубликаты строк

Примеры использования:

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

Первый символ в образце команды sed, табуляция, что-бы написать его в консоли, используйте ctrl-v ctrl-tab.

Системная оболочка bash

Перенаправления ввода/вывода и пайпы в *bash* и *sh*:

```
# 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                       # Пайп stdout на вход команды cmd2
# cmd1 2>&1 | cmd2                  # Пайп stdout и stderr команде cmd2
```

Настройка оболочки в файле конфигурации *~/.bashrc* (так-же может быть *~/.bash_profile*).

```
# in .bashrc
bind '"\e[A"':history-search-backward # Использовать клавиши "вверх" и "вниз" для
поиска.
bind '"\e[B"':history-search-forward  # История введенных команд
set -o emacs                        # Установить emacs режим в bash (см. ниже)
set bell-style visible              # Не подавать звуковой сигна, инверировать
цвета
# Настройка приглашения строки [user@host]/path/todir>
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\[\033[1;30m\]>\[\033[0m\]"
```

Что-бы посмотреть используемые псевдонимы (alias) команд, используйте команду *alias*

```
alias  ls='ls -aF'                    # Добавить индикатор (один из */=>@|)
alias  ll='ls -aFls'                  # Лмстинг файлов и каталогов
alias  la='ls -all'
alias ..='cd ..'
alias ...='cd ../..'
export HISTFILESIZE=5000              # Увеличить историю
export CLICOLOR=1                     # Использовать цвета (если возможно)
export LSCOLORS=ExGxFxdxCxDxDxBxBxExEx
```

Системная оболочка tcsh

Перенаправления и пайпы для tcsh и csh (> и >> действуют как в sh):

```
# cmd >& file                         # Перенаправить stdout и stderr в файл.
# cmd >>& file                        # Добавить вывод stdout и stderr в конец файла.
# cmd1 | cmd2                         # Перенаправить stdout на вход cmd2
# cmd1 |& cmd2                        # Перенаправить stdout и stderr на вход cmd2
```

Настройка оболочки *csh (tcsh)*, делается в файле *~/.cshrc*, перезагрузить можно командой *source .cshrc*. Примеры:

```
# in .cshrc
alias  ls      'ls -aF'
alias  ll      'ls -aFls'
alias  la      'ls -all'
alias  ..      'cd ..'
alias  ...     'cd ../..'
set   prompt    = "%B%n%b@%B%m%b%/> " # Приглашение командной строки:
user@host/path/todir>
set   history   =  5000
set   savehist  = (6000 merge)
set   autolist                        # Список возможного дописывания команд по
нажатию Tab
set   visiblebell                     # Не выдавать звуковой сигнал, инвертировать
цвета
```

Цепляем клавиши и цвета:

```
bindkey -e     Select Emacs bindings  # Использовать сочетания клавиш emacs для
редактирования командной строки
bindkey -k up history-search-backward # Использовать и для поиска
bindkey -k down history-search-forward
setenv CLICOLOR 1                     # Использовать цвета (если возможно)
setenv LSCOLORS ExGxFxdxCxDxDxBxBxExEx
```

Режим *emacs*, включает горячие клавиши в стиле *emacs*, для редактирования командной строки. Это очень удобно и не только для пользователей emacs. Часто используемые команды:

- C-a — Переместить курсор в начало строки
- C-e — Переместить курсор в конец строки
- M-b — Переместить курсор на одно слово назад
- M-f — Переместить курсор на одно слово вперед
- M-d — Удалить слово слева
- C-w — Удалить слово справа

- C-u — Удалить все до курсора
- C-k — Удалить все после курсора
- C-y — Вставить в строку последнее вырезанное
- C-_ — Назад

Справка: C- = зажать Ctrl, M- = зажать *meta* (обычно клавиша Alt или Esc).

Печать

```
# lpr exemple.ps                      # Печать на принтер по-умолчанию
# export PRINTER=hp4600               # Сменить принтер по-умолчанию
# lpr -Php4500 #2 unixtoolbox.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.
```

Некоторые принтеры не поддерживают postscript и могут выводить на печать мусор. Исправляется следующим образом:

```
# gs -dSAFER -dNOPAUSE -sDEVICE=deskjet -sOutputFile=\|lpr file.pdf
```