UNIX Куфарче

Този документ е колекция от Unix/Linux/BSD команди и задачи, които са полезни за ИТ работа или напреднали потребителеи. Това е практически гайд, съдържащ обяснения, които да указват на читателя/ката какво прави.

Unix Куфарче ревизия 11
Последната версия на този документ може да бъде намерена на http://cb.vu/unixtoolbox.xhtml. Заменете .xhtml във връзката с .pdf за PDF версията и с .book.pdf за книжната версия. На дуплекс принтер книжката ще създаде малка книга готова за обвързване.

Съобщения за грешки и коментари са добре дошли - c@cb.vu Colin Barschel.

1 Система

Зареждане на ядрото и информация за системата

```
# uname -a                          # Дава версията на ядрото (и BSD версията)
# cat /etc/SuSE-release             # Дава версията на SuSE
# cat /etc/debian_version           # Дава Debian версията
```

Използвай /etc/DISTR-release с DISTR= lsb (Ubuntu), redhat, gentoo, mandrake, sun (Solaris), и т.н.

```
# uptime                            # Показва колко време системата е работила + зареждане
# hostname                          # име на хоста на системата
# hostname -i                       # Показва IP адреса на системата.
# man hier                          # Описание на йерархията на файловата система
# last reboot                       # Показва история на рестартиранията на системата
```

1.1 Информация за хардуера

Хардуер открит от ядрото

```
# dmesg                             # Открит хардуер и съобщения при буут
# lsdev                             # информация за инсталиран хардуер
# dd if=/dev/mem bs=1k skip=768 count=256 2>/dev/null | strings -n 8 # Чете BIOS
```

Линукс

```
# cat /proc/cpuinfo                 # Модел на процесора
# cat /proc/meminfo                 # Хардуерна памет
# grep MemTotal /proc/meminfo       # Показва физическата памет
# watch -n1 'cat /proc/interrupts'  # Следи постоянно за променливи прекъсвания
# free -m                           # Използвана и свободна памет (-m за MB)
# cat /proc/devices                 # Конфигурирани устройства
# lspci -tv                         # Показва PCI устройства
# lsusb -tv                         # Показва USB устройства
# lshal                             # Показва списък от всички устройства и техните настройки
# dmidecode                         # Показва DMI/SMBIOS: хардуер информация от BIOS
```

FreeBSD

```
# sysctl hw.model                   # Модел на процесора
# sysctl hw                         # Дава пълна информация за хардуера
# sysctl vm                         # Използвана памет
# dmesg | grep "real mem"           # Хардуерна памет
# sysctl -a | grep mem              # Памет на настройките на ядрото и информация
# sysctl dev                        # Конфигурирани устройства
# pciconf -l -cv                    # Показва PCI устройства
# usbdevs -v                        # Показва USB устройства
# atacontrol list                   # Показва ATA устройства
```

1.2 Зареждане, статистики и съобщения

Следващите команди са полезни за да откриете какво се случва в системата.

```
# top                               # Показва и обновява топ процесите на процесора
# mpstat 1                          # Показва статистики свързани с процесори
# vmstat 2                          # Показва статистики на виртуалната памет
# iostat 2                          # Показва I/O статистики (2-секундни интервали)
# systat -vmstat 1                  # BSD обобщение на системните статистики (1-секунден интерва
```

23 Онлайн помощ

23.1 Документация

Linux Документация	en.tldp.org
Linux Man Страници	www.linuxmanpages.com
Linux директория с командите	www.oreillynet.com/linux/cmd
Linux doc man howtos	linux.die.net
FreeBSD Handbook	www.freebsd.org/handbook
FreeBSD Man Страници	www.freebsd.org/cgi/man.cgi
FreeBSD потребителско wiki	www.freebsdwiki.net
Solaris Man Страници	docs.sun.com/app/docs/coll/40.10

23.2 Други Unix/Linux справочници

Rosetta Stone for Unix	bhami.com/rosetta.html (a Unix команден преводач)
Unix гид	unixguide.net/unixguide.shtml
Linux команден списък	www.linuxguide.it/commands_list.php
Къс Linux справочник	www.pixelbeat.org/cmdline.html

Това е всичко!


```
# systat -tcp 1                     # BSD tcp връзки (пробрай също -ip)
# systat -netstat 1                 # BSD активни мрежови връзки
# systat -ifstat 1                  # BSD мрежови трафик през активни интерфейси
# systat -iostat 1                  # BSD CPU and and disk throughput
# tail -n 500 /var/log/messages     # Последните 500 съобщения на ядро/сислог
# tail /var/log/warn                # Системни предупрежения виж syslog.conf
```

1.3 Потребители

```
# id                                # Показва активното потребителско id с име и група
# last                              # Показва последните влизания в системата
# who                               # Показва кой е влязъл в системта
# groupadd admin                    # Добавя група "admin" и потребител colin (Linux/Solaris)
# useradd -c "Colin Barschel" -g admin -m colin
# userdel colin                     # Изтрива потребител colin (Linux/Solaris)
# adduser joe                       # FreeBSD добавя потребител joe (interactive)
# rmuser joe                        # FreeBSD изтрива потебител joe (interactive)
# pw groupadd admin                 # Използва pw в FreeBSD
# pw groupmod admin -m newmember    # Добавя нов член в група
# pw useradd colin -c "Colin Barschel" -g admin -m -s /bin/tcsh
# pw userdel colin; pw groupdel admin
```

Криптирани пароли са записани в /etc/shadow за Linux и Solaris и /etc/master.passwd за FreeBSD. Ако master.passwd е променен ръчно (да речем заради изтриване на парола), стартирайте `# pwd_mkdb -p master.passwd` за пренареждане на базата данни.

За да изключите временно влизането в цялата система (за всички потребител без root) използвайте nologin. Съобещението ще бъде показано в nologin.

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

1.4 Граници

Някои приложения изискват по-високи граници за отваряне на файлове и гнезда (като прокси уеб сървър, база данни). Границите по подразбиране обикновено са твърде ниски.

Линукс

На шел/скрипт

Границите на шела се управляват от `ulimit`. Статуса се проверява с `ulimit -a`. На пример, за да промените границата за отваряне на файлове от 1024 до 10240 изпълнете:

```
# ulimit -n 10240                   # Това е валидно само в шела
```

`ulimit` командата може да се използва в скрипт, за да промени границите само за скрипта.

На потребител/процес

Влезли потребители и програми могат да бъдат конфигурирани в `/etc/security/limits.conf`. На пример:

```
# cat /etc/security/limits.conf
*   hard    nproc   250             # Ограничава потребителските процеси
asterisk hard nofile 409600         # Ограничава програмите в отварянето на файлове
```

За цялата система

Ограничения на ядрото се слагат със sysctl. Постоянни ограничения са записани в `/etc/sysctl.conf`.

```
# sysctl -a                          # Показва всички системни ограниченияView all system limits
# sysctl fs.file-max                 # Показва максималното ограничение за отваряне на файлове
# sysctl fs.file-max=102400          # Променя максималното ограничение за отваряне на фйалове
# cat /etc/sysctl.conf
fs.file-max=102400                   # Постоянен запис в sysctl.conf
# cat /proc/sys/fs/file-nr           # Колко описания на файлове се използват
```

FreeBSD

На шел/скрипт

Използвайте командата `limits` в csh или tcsh, или както в Linux, използвайте `ulimit` в sh или bash шел.

На потребител/процес

Ограниченията по подразбиране при влизане са записани в `/etc/login.conf`. Неограничената стойност е все още ограничена от системната максимална стойност.

За цялата система

Ограниченията на ядрото също се слагат със sysctl. Постоянни връзки са записани в `/etc/sysctl.conf` или `/boot/loader.conf`. Синтаксисът е същият като при Linux но ключовете са различни.

```
# sysctl -a                          # Показва всички системни ограничения
# sysctl kern.maxfiles=XXXX          # Максимален брой на файловите описания
kern.ipc.nmbclusters=32768           # Постоянна връзка в /etc/sysctl.conf
kern.maxfiles=65536                  # Типични стойности за Squid
kern.maxfilesperproc=32768
kern.ipc.somaxconn=8192              # TCP опашка. По-добре за apache/sendmail
# sysctl kern.openfiles              # Колко файлови описания се използват
# sysctl kern.ipc.numopensockets     # Колко отворени гнезда се използват
```

Виж FreeBSD handbook Chapter 11 за подробности.

Solaris

Следващите стойности в `/etc/system` ще увеличат максималните файлови описания на процес:

```
set rlim_fd_max = 4096               # Твърда граница на файлово описания за един процес
set rlim_fd_cur = 1024               # Мека граница на файлово описания за един процес
```

1.5 Нива на работа

Linux

Веднъж заредено, ядрото стартира `init`, който после стартира `rc`, който пък стартира всички скриптове принадлежащи към нивото на работа. Скриптовете са записани в /etc/init.d и имат връзки в /etc/rc.d/rcN.d с N номера на нивото на работа.
Нивото на зареждане по подразбиране е конфигурирано в /etc/inittab. Обикновено е 3 или 5:

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

Актуалното ниво на зареждане (списъкът е показан отдолу) може да бъде променен с `init`. На пример, за да го промените от 3 на 5:

```
# init 5                             # Въвежда ниво на работа 5
```


IPv4.cpp:

```
#include "IPv4.h"
#include <string>
#include <sstream>
using namespace std;                                  // използване мястото на имената
using namespace GenericUtils;

IPv4::IPv4() {}                                       // главен конструктор/деструктор
IPv4::~IPv4() {}
string IPv4::IPint_to_IPquad(unsigned long ip) {      // реализация на члена
    ostringstream ipstr;                              // използване на поток
    ipstr << ((ip &0xff000000) >> 24)                 // побитово десен шифт
          << "." << ((ip &0x00ff0000) >> 16)
          << "." << ((ip &0x0000ff00) >> 8)
          << "." << ((ip &0x000000ff));
    return ipstr.str();
}
```

The program simplecpp.cpp

```
#include "IPv4.h"
#include <iostream>
#include <string>
using namespace std;

int main (int argc, char* argv[]) {
    string ipstr;                                     // дефиниране на променливите
    unsigned long ipint = 1347861486;                 // IP целочислен вид
    GenericUtils::IPv4 iputils;                       // създаване на обект от класа
    ipstr = iputils.IPint_to_IPquad(ipint);           // извикване на члена на класа
    cout << ipint << " = " << ipstr << endl;          // печат на резултата

    return 0;
}
```

Компилация и изпълнение с:

```
# g++ -c IPv4.cpp simplecpp.cpp                 # Компилиране в обекти
# g++ IPv4.o simplecpp.o -o simplecpp.exe       # Свързване обектите в крайна изпълнима програма
# ./simplecpp.exe
1347861486 = 80.86.187.238
```

Използвайте `ldd` за проверка кои библиотеки са използвани от програмата и къде се намират. Командата също се използва ако споделена библиотека липсва или програмата е статична.

```
# ldd /sbin/ifconfig
```

22.5 Прост Makefile

Следният минимален Makefile за мулти-сорс програма е показан тук. Редовете с инструкции *трябва да започват с таб*! Обратната наклонена "\" може да се използва за разделяне на дълги редове.

```
CC = g++
CFLAGS = -O
OBJS = IPv4.o simplecpp.o

simplecpp: ${OBJS}
        ${CC} -o simplecpp ${CFLAGS} ${OBJS}
clean:
        rm -f ${TARGET} ${OBJS}
```

22.2 C пример

Минимална програма simple.c:

```
#include <stdio.h>
main() {
    int number=42;
    printf("The answer is %i\n", number);
}
```

Комплиране:

```
# gcc simple.c -o simple
# ./simple
The answer is 42
```

22.3 C++ база

```
*pointer                                    // Обект посочен с указател
&obj                                        // Адрес на обекта obj
obj.x                                       // Член x от клас obj (обект obj)
pobj->x                                     // Член x от клас посочен чрез pobj
                                            // (*pobj).x и pobj->x са едно и също
```

22.4 C++ пример

Като по-реална програма на C++, нека направим клас във неговия собствен хедър (IPv4.h) и реализация (IPv4.cpp) и да направим програма, която използва класове. Класът има член конвертиращ IP адрес в целочислен формат към четиригрупов формат. Това е минимална c++ програма с клас и мулти-сорс компилация.

IPv4 клас

IPv4.h:

```
#ifndef IPV4_H
#define IPV4_H
#include <string>

namespace GenericUtils {                        // създаване на масто на имената
class IPv4 {                                    // дефиниция на класа
public:
    IPv4();
    ~IPv4();
    std::string IPint_to_IPquad(unsigned long ip);// интервейс на члена на класа
};
} //място имена GenericUtils
#endif // IPV4_H
```


0 Изключи и задръж
1 Потребителски режим (също S)
2 Многопотребителски режим с мрежа
3 Многопотребителски режим без мрежа
5 Многопотребителски режим с X
6 Рестартира

Използвайте `chkconfig`, за да конфигурирате програми, които стартират при зареждане в ниво на работа.

```
# chkconfig --list                    # Покажи всички init скриптове
# chkconfig --list sshd               # Рапортира статуса на sshd
# chkconfig sshd --level 35 on        # Конфигурира sshd за нива 3 и 5
# chkconfig sshd off                  # Забрани sshd за всички нива на работа
```

Дебиан и дебиан-базираните дистрибуции като Убунту и Кнопикс използват командата `update-rc.d` за да контролират скриптовете за нива на работа. По подразбиране е да започне в 2,3,4 и 5 и да изключи в 0,1 и 6.

```
# update-rc.d sshd defaults           # Активира sshd със стандартните нива на работа
# update-rc.d sshd start 20 2 3 4 5 . stop 20 0 1 6 .  # С определени аргументи
# update-rc.d -f sshd remove          # Забранява sshd за всички нива на зареждане
# shutdown -h now (or # poweroff)     # Изключва и задържа системата
```

FreeBSD

BSD обръщенията при пускане са различават от SysV, там няма нива на работа. Последното положение при пускане (отделен потребител, с или без X) е конфигурирано в`/etc/ttys`. Всички ОС скриптове се намират в `/etc/rc.d/` и в `/usr/local/etc/rc.d/` за трети приложения. Активирането на услуга е конфигурирано в `/etc/rc.conf` и `/etc/rc.conf.local`. Поведението по подразбиране е конфигурирано в `/etc/defaults/rc.conf`. Скриптовете отговарят поне на старт|стоп|статус.

```
# /etc/rc.d/sshd status
sshd is running as pid 552.
# shutdown now                        # Премини в потребителски режим
# exit                                # Премини в многопотребителски режим
# shutdown -p now                     # Изключи и задръж системата
# shutdown -r now                     # Рестартирай
```

Процесът `init` може също да бъде достигнат от следните нива. Например `# init 6` за рестарт.

0 Задръж и изключи захранването (сигнал `USR2`)
1 Премини в потребителски режим (сигнал `TERM`)
6 Рестартирай машината (сигнал `INT`)
c Блокирай следващите влизания (сигнал `TSTP`)
q Сканирай ttys(5) файла (сигнал `HUP`)

1.6 Промени root паролата

Linux метод 1

При зареждането на системата (lilo или grub), въведете следната опция на стартиране:

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

Ядрото ще прикачи root дяла и `init` ще стартира борн-шела вместо `rc` в нивото на зареждане. Използвайте командата `passwd` в конзолата, за да промените паролата и после рестартирайте. Забравете потребителския режим, защото паролата ви трябва тук.
Ако след рестарт, root дяла е прикачен само за четене, го прикачете отново rw:

```
# mount -o remount,rw /
# passwd                              # или изтрийте root паролата (/etc/shadow)
```

```
# sync; mount -o remount,ro /        # синхронизира преди повторно прикачване само за четене
# reboot
```

FreeBSD и Linux метод 2

FreeBSD няма да ви позовли да се измъкнете само с прост init трик. Решението е да прикачите root дяла от друга ОС (като спасителното CD) и да промените паролата на диска.

- Стартирайте живо CD или инсталационно CD в спасителен режим, който ще ви даде шел.
- Намерете root дяла с fdisk н.пр. fdisk /dev/sda
- Монтирайте го и използвайте chroot:

```
# mount -o rw /dev/ad4s3a /mnt
# chroot /mnt                        # chroot в /mnt
# passwd
# reboot
```

Като алтернатива във FreeBSD, стартирайте потребителски режим, монтирайте отново / rw и използвайте passwd.

```
# mount -u /; mount -a               # ще прикачи / rw
# passwd
# reboot
```

1.7 Модули на ядрото

Linux

```
# lsmod                              # Показва всички модули заредени в ядрото
# modprobe isdn                      # За зареждане на модул (тук isdn)
```

FreeBSD

```
# kldstat                            # Показва всички модули заредени в ядрото
# kldload crypto                     # За зареждане на модул (тук crypto)
```

1.8 Компилиране на ядрото

Linux

```
# cd /usr/src/linux
# make mrproper                      # Изкчиства всичко, включително конфигурационни файлове
# make oldconfig                     # Създава нов конфигурационен файл от настоящото ядро
# make menuconfig                    # или xconfig (Qt), или gconfig (GTK)
# make                               # Създава компресирана снимка на ядрото
# make modules                       # Компилира модулите
# make modules_install               # Инсталира модулите
# make install                       # Инсталира ядрото
# reboot
```

FreeBSD

За да модифицирате и препостроите ядрото копирайте общия конфигурационен файл с ново име и го редактирайте при нужда. Всъщност е възможно и да промените файла `GENERIC` директно.

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



```
#!/bin/sh
# Този скрипт създава книжка в pdf формат готова за печат на двоен принтер
if [ $# -ne 1 ]; then                          # Проверка на аргумента
  echo 1>&2 "Usage: $0 HtmlFile"
  exit 1                                       # не нулев изход при грешка
fi

file=$1                                        # Задаване на файлово име
fname=${file%.*}                               # Вземане само името на файла
fext=${file#*.}                                # Вземане разширението на името

prince $file -o $fname.pdf                     # от www.princexml.com
pdftops -paper A4 -noshrink $fname.pdf $fname.ps # създаване на постскрипт брошура
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
                                               # използване на #a4 и #None във Windows!
exit 0                                         # exit 0 означава успешно
```

21.3 Някои sed команди

```
sed 's/string1/string2/g'                      # Замяна string1 със string2
sed -i 's/wroong/wrong/g' *.txt                # Замяна неправилна дума с g
sed 's/\(.*\)1/\12/g'                          # Промяна anystring1 със anystring2
sed '/<p>/,/<\/p>/d' t.xhtml                   # Изтриване редове започващи с <p>
                                               # и завършващи с </p>
sed '/ *#/d; /^ *$/d'                          # Изтриване на коментари и празни редове
sed 's/[ \t]*$//'                              # Изтриване на крайните интервали (използване на tab
sed 's/^[ \t]*//;s/[ \t]*$//'                  # Изтриване на началните и крайните интервали
sed 's/[^*]/[&]/'                              # Вмъкване на първия символ в [] top->[t]op
```

21.4 Полезни команди

```
sort -t. -k1,1n -k2,2n -k3,3n -k4,4n           # Сортиране IPv4 ip адреси
echo 'Test' | tr '[:lower:]' '[:upper:]'       # Смяна на големи/малки букви
echo foo.bar | cut -d . -f 1                   # Връща foo
PID=$(ps | grep script.sh | grep bin | awk '{print $1}')     # PID на работещия скрипт
PID=$(ps axww | grep [p]ing | awk '{print $1}')              # PID на ping (без grep pid)
IP=$(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       # Променен файл?
cat /etc/master.passwd | grep -v root | grep -v \*: | awk -F":" \ # Създаване http парола
'{ printf("%s:%s\n", $1, $2) }' > /usr/local/etc/apache2/passwd

testuser=$(cat /usr/local/etc/apache2/passwd | grep -v \     # Проверка на потребителя в passwd
root | grep -v \*: | awk -F":" '{ printf("%s\n", $1) }' | grep ^user$)
```

22 Програмиране

22.1 C база

```
strcpy(newstr,str)                             /* копиране str в newstr */
expr1 ? expr2 : expr3                          /* ако (expr1) expr2 иначе expr3 */
x = (y > z) ? y : z;                           /* ако (y > z) x = y; иначе x = z; */
int a[]={0,1,2};                               /* Инициализиран масив (или a[3]={0,1,2}; */
int a[2][3]={{1,2,3},{4,5,6}};                 /* Масив от масив от цели числа */
int i = 12345;                                 /* Конвертиране на i в символен низ */
char str[10];
sprintf(str, "%d", i);
```

Специални променливи

```
$$                                      #ID на текущия процес
$?                                      # статус на изхода от последната команда
  command
  if [ $? != 0 ]; then
    echo "command failed"
  fi
mypath=`pwd`
mypath=${mypath}/file.txt
echo ${mypath##*/}                      # Показване само на името на файл
echo ${mypath%%.*}                      # Пълен път без разширение
var2=${var:=string}                     # Използвай var ако е установено, иначе използвай из
                                        # Установяване на израз към var и после към 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 е първия аргумент на функция
}
myfunction "txt"
```

Генериране на файл

```
MYHOME=/home/colin
cat > testhome.sh << _EOF
# Всичко това отива във файла testhome.sh
if [ -d "$MYHOME" ] ; then
    echo $MYHOME exists
else
    echo $MYHOME does not exist
fi
_EOF
sh testhome.sh
```

21.2 Bourne пример за скрипт

Като пример, скриптът използван за създаване на PDF брошура от този xhtml документ:

За да построите отново цялата ОС:

```
# make buildworld                       # Построява цялата ОС, но без ядрото
# make buildkernel                      # Използва KERNCONF както горе ако е подходящо
# make installkernel
# reboot
# mergemaster -p                        # Сравнява само файлове познати като крайно необходими
# make installworld
# mergemaster                           # Подновява цялата конфигурация и други файлове
# reboot
```

За малки промени в изхода, понякога е достатъчна и кратката версия:

```
# make kernel world                     # Компилира и инсталира и двете - ядро и ОС
# mergemaster
# reboot
```

2 Процеси

2.1 Показване на ПОУ (PID = Обозначаващ указател за процес)

Всеки процес има уникален номер, ПОУ-то. Списък на всички работещи процеси се изкарва с `ps`.

```
# ps -auxefw                            # Extensive list of all running process
```

Обаче по-типично е използването с pipe или с `pgrep`:

```
# ps axww | grep cron
  586  ??  Is     0:01.48 /usr/sbin/cron -s
# pgrep -l sshd                         # Намира ПОУ на процес по (част от) име
# fuser -va 22/tcp                      # Показва процесите, ползващи порт 22
# fuser -va /home                       # Показва процеси използващи достъп до /home дяла
# strace df                             # Проследява системни обаждания и сигнали
# truss df                              # Същото като горното на FreeBSD/Solaris/Unixware
# history | tail -50                    # Показва последните 50 използвани команди
```

2.2 Приоритет

Променя приоритета на работещ процес с `renice`. Негативни числа имат по-висок приоритет, най-малкото е -20 и "nice" има позитивни стойности.

```
# renice -5 586                         # По-силен приоритет
586: old priority 0, new priority -5
```

Стартиране на процес с различен приоритет с `nice`. Позитивно е "nice" или слабо, негативно е силно засенчващ приоритет. Уверете се, дали се използва `/usr/bin/nice` или вградения в шела (проверка с `# which nice`).

```
# nice -n -5 top                        # Силен приоритет (/usr/bin/nice)
# nice -n 5 top                         # Слаб приеоритет (/usr/bin/nice)
# nice +5 top                           # tcsh вграден nice (същото като горе!)
```

2.3 Фон/Преден План

Когато процес е стартиран от шела, той може да бъде преместен на задел план или обратно на преден план със [Ctrl]-[Z] (^Z), `bg` и `fg`. На пример стартирайте два процеса във фона, покажете ги с `jobs` и ги преместете на преден план.

```
# ping cb.vu > ping.log
^Z                                  # е прекъсната (спряна) с [Ctrl]-[Z]
# bg                                # слага във фона и продължава работа
# jobs -l                           # Показва процесите на заден план
[1]  - 36232 Running                      ping cb.vu > ping.log
[2]  + 36233 Suspended (tty output)       top
# fg %2                             # Премества процес 2 на преден план
```

Използвайте `nohup` за стартиране на процес, който продължава да работи, когато шел-а бъде затворен (имунитет против прекъсвания).

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

2.4 Топ

Програмата `top` показва работеща информация за процеси.

```
# top
```

Докато top работи натиснете клавиша h за помощ. Полезни ключове са:

- **u [user name]** За показване ан процеси принадлежащи на потребител. Използвайте + или шпация, за да видите всички потребители
- **k [pid]** Убива всички процеси с ПОУ.
- **1** за да покаже всички статистики на процесора (само за Линукс)
- **R** Toggle normal/reverse sort.

2.5 Сигнали/Kill

Прекъсване или изпращане на сигнал с `kill` или `killall`.

```
# ping -i 60 cb.vu > ping.log &
[1] 4712
# kill -s TERM 4712                 # същото като kill -15 4712
# killall -1 httpd                  # Прекъсва(убува) HUP процеси по точно име
# pkill -9 http                     # Прекъсва TERM процеси по (част от) име
# pkill -TERM -u www                # Прекъсва TERM процеси принаглежащи на www
# fuser -k -TERM -m /home           # Прекъсва вески процес с достъп до /home (до стойност)
```

Важни сигнали са:

1	`HUP` (затвори)
2	`INT` (прекъсни)
3	`QUIT` (излез)
9	`KILL` (non-catchable, non-ignorable kill)
15	`TERM` (software termination signal)

3 Файлова система

Информация за диска (p9) | Зареждане (p9) | Използване на диска (p9) | Отворени файлове (p9) | Прикачване/Преприкачване (p10) | Прикачване на SMB (p11) | Прикачване на image (p11) | Запис на ISO (p12) | Създай image (p13) | Диск памет (p13) | Производителност на диска (p14)

3.1 Ограничения

Промяна на ограничения и собственост с `chmod` и `chown`. Umask по подразбиране може да бъде променен за всички потребители в /etc/profile за Linux или /etc/login.conf за FreeBSD. umask по подразбиране обикновено е 022. umsak е извадена от 777, така umask 022 се променя в ограничение 755.


```
# в .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%/> " # като user@host/path/todir>
set   history   =  5000
set   savehist  = ( 6000 merge )
set   autolist                        # Показване яъзможните завършвания с таб
set   visiblebell                     # Без бип, инверсия на цветовете
```

```
# Bindkey и цвят
bindkey -e     Select Emacs bindings  # Използване на emacs клавиши за редактиране на команден ре
bindkey -k up history-search-backward # Използване нагоре и надолу за търсене
bindkey -k down history-search-forward
setenv CLICOLOR 1                     # С цвят (ако може)
setenv LSCOLORS ExGxFxdxCxDxDxBxBxExEx
```

emacs модът позволява ползване на emacs клавиши за промяна на команден ред. Изключително полезно е (не само за emacs потребители). Най-използваните команди са:

C-a	Преместване на курсора в началото на линията
C-e	Преместване на курсора в края на линията
M-b	Преместване на курсора назад една дума
M-f	Преместване на курсора напред една дума
M-d	Отрязване следващата дума
C-w	Отрязване последната дума
C-u	Отрязване всичко преди курсора
C-k	Отрязване всичко след курсора (до края на реда)
C-y	Поставяне последното за отрязване (просто поставяне)
C-_	Отмени

Note: C- = задръж control, M- = задръж мета (обикновено alt или escape бутона).

21 Скриптове

База (p49) | Примерен скрипт (p50) | sed/полезни команди (p51)

Bourne шела (/bin/sh) присъства във всички Unix инсталации и скриптовете написани на него са (доста) преносими; `man 1 sh` е добър справочник.

21.1 База

Променливи и аргументи

Задаване с variable=стойност и проверка с $variable

```
MESSAGE="Hello World"                 # Задаване на израз
PI=3.1415                             # Задаване на стойност
N=8
TWON=`expr $N * 2`                    # Аритметичен израз (само цели числа)
TWON=$(($N * 2))                      # друг синтакс
TWOPI=`echo "$PI * 2" | bc -l`        # Използване на bc за операции с плаваща запетая
ZERO=`echo "c($PI/4)-sqrt(2)/2" | bc -l`
```

Аргументите в команден ред са

```
$0, $1, $2, ...                       # $0 е самата команда
$#                                    # Броя аргументи
$*                                    # Всички аргументи (също $@)
```

20 Обвивки (SHELLS)

Повечето Linux дистрибуции използват bash shell, докато BSD-тата използват tcsh, bourne shell се използва само за скриптове. Филтрите за много полезни и могат да се обединяват:

- `grep` Съвпадане на израз
- `sed` Търсене и замяна на израз или дума
- `cut` Отпечатване на специфични колони от маркер
- `sort` Азбучно и цифрово сортиране
- `uniq` Премахване дублираните линии от файл

За пример всички използвани заедно:

```
# 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
```

Първият символ в sed израза е таб. За изписване таб в конзола, използвайте ctrl-v ctrl-tab.

20.1 bash

Пренасочване и обединяване за bash и sh:

```
# cmd 1> file                          # Пренасочване stdout към файл.
# cmd 2> file                          # Пренасочване stderr към файл.
# cmd 1>> file                         # пренасочване и добавяне stdout към файл.
# cmd &> file                          # Пренасочване двете stdout и stderr към файл.
# cmd >file 2>&1                       # Redirects stderr to stdout and then to file.
# cmd1 | cmd2                          # обединяване stdout с cmd2
# cmd1 2>&1 | cmd2                     # обединяване stdout и stderr с cmd2
```

Променете вашата конфигурация в ~/.bashrc (или ~/.bash_profile). Следните команди могат да са полезни, презаредете чрез ". .bashrc".

```
# в .bashrc
bind '"\e[A"':history-search-backward # Използване нагоре и надолу за търсене
bind '"\e[B"':history-search-forward  # историята Безценно!
set -o emacs                          # Настройка на emacs мод в bash (виж надолу)
set bell-style visible                # Без бип, инверсия на цветовете
    # Настройка на промпт във вид например [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\[\033[1;30m\]>\[\033[0m\]"
```

```
# За проверка на активните замени, просто напишете alias
alias  ls='ls -aF'                    # Добавяне на индикатор (един от */=>@|)
alias  ll='ls -aFls'                  # лист
alias  la='ls -all'
alias ..='cd ..'
alias ...='cd ../..'
export HISTFILESIZE=5000              # По-голяма история
export CLICOLOR=1                     # С цвят (ако може)
export LSCOLORS=ExGxFxdxCxDxDxBxBxExEx
```

20.2 tcsh

Пренасочванията и обединенията за tcsh и csh (просто > и >> за също като sh):

```
# cmd >& file                          # Пренасочване на двете stdout и stderr във файл.
# cmd >>& file                         # Добавяне на двете stdout и stderr във файл.
# cmd1 | cmd2                          # обединение stdout с cmd2
# cmd1 |& cmd2                         # оединение stdout и stderr в cmd2
```

Настойките на csh/tcsh са в `~/.cshrc`, презаредете с "source .cshrc". Примери:


```
1 --x execute                         # Режим 764 = exec/read/write | read/write | read
2 -w- write                           # За:       |--  Owner  --|   |- Group-|   |Oth|
4 r-- read
  ugo=a                               u=потребител, g=група, o=други, a=всеки
```

```
# chmod [OPTION] MODE[,MODE] FILE     # MODE е във формата [ugoa]*([-+=]([rwxXst]))
# chmod 640 /var/log/maillog          # Ограничава лог-а на -rw-r-----
# chmod u=rw,g=r,o= /var/log/maillog # Същото като горе
# chmod -R o-r /home/*                # Рекурсивно премахва четенето за всички потребители
# chmod u+s /path/to/prog             # Слага SUID бит на изпълними (трябва да знаете какво правит
# find / -perm -u+s -print            # Открива всички програми със SUID бит
# chown user:group /path/to/file      # Променя потребителската и груповата собственост над файл
# chgrp group /path/to/file           # Променя собствеността на файл за групата
```

3.2 Информация за диск

```
# diskinfo -v /dev/ad2                # Информация за диск (сектор/размер) FreeBSD
# hdparm -I /dev/sda                  # Информация за IDE/ATA диск (Linux)
# fdisk /dev/ad2                      # Показва и манипулира таблицата на дяла (partition table)
# smartctl -a /dev/ad2                # Показва SMART информацията на диска
```

3.3 Зареждане

FreeBSD

За да заредите старо ядро, ако новото не се зарежда, спрете зареждането по време на отброяването.

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

3.4 Системни точки на прикачване/Използване на диска

```
# mount | column -t                  # Показва прикаченитеите файлови системи в системата
# df                                  # показва свободното място на диска и прикачените устройства
# cat /proc/partitions                # Показва всички регистрирани дялове (Linux)
```

Използване на диска

```
# du -sh *                            # Размер на директориите като списък
# du -csh                             # Общ размер на директорията за настоящата директория (тази,
# du -ks * | sort -n -r               # Сортира всичко по размер в килобайти
# ls -lSr                             # Показва файлове, най-големия последен
```

3.5 Кой кои файлове е отворил

Това е полезно за откриване на това - кой файл блокира дял, която трябва да бъде откачен и дава типична грешка за:

```
# umount /home/
umount: unmount of /home              # umount е невъзможен, защото файл заключва home
   failed: Device busy
```

FreeBSD и повечето Unixes

```
# fstat -f /home                      # за точка на прикачане
# fstat -p PID                        # за приложение с PID
# fstat -u user                       # за потребителско име
```

Открива отворен лог-файл (или отворени файлове), да речем за 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
```

Файлът с 212042 е единственият файл във /var:

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

Linux

Открива отворени файлове в точка на прикачване с `fuser` или `lsof`:

```
# fuser -m /home                    # Показва процеси с достъп до /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)
```

Относно приложение:

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

Относно единичен файл:

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

3.6 Прикачване/преприкачване на файлова система

На пример cdrom. Ако е в списъка /etc/fstab:

```
# mount /cdrom
```

Или намира устройство в /dev/ или с dmesg

FreeBSD

```
# mount -v -t cd9660 /dev/cd0c /mnt  # cd-устройство
# mount_cd9660 /dev/wcd0c /cdrom     # друг метод
# mount -v -t msdos /dev/fd0c /mnt   # флопи
```

Декларирано в /etc/fstab:

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

За да позволите на потребителите да го правят:

```
# sysctl vfs.usermount=1  # Или добавете реда "vfs.usermount=1" в /etc/sysctl.conf
```

Linux

```
# mount -t auto /dev/cdrom /mnt/cdrom   # типична команда за прикачване на cdrom
# mount /dev/hdc -t iso9660 -r /cdrom   # типично IDE
# mount /dev/sdc0 -t iso9660 -r /cdrom  # типично SCSI
```


19.2 FreeBSD настройка

Инструментите за квота за част от базовата система, но ядрото има нужда от опция. Ако не е поставена добавете я и прекомпилирайте ядрото.

```
опции QUOTA
```

Като в Linux, добавете квота във fstab опциите (userquota, not usrquota):

```
/dev/ad0s1d    /home    ufs     rw,noatime,userquota    2  2
# mount /home                          # за премонтиране на дял
```

разрешаване на дисковите квоти в /etc/rc.conf и стартиране.

```
# grep quotas /etc/rc.conf
enable_quotas="YES"                    # включване на квота при старт (или NO).
check_quotas="YES"                     # Проверка на квота при старт (или NO).
# /etc/rc.d/quota start
```

19.3 Задаване на лимити

Квотите не са лимитирани по подразбиране (установени са на 0). Лимитите се установяват с `edquota` за потребителите. Квотата може да бъде повторена за много потребители. Файловата структура е различна при реализациите на квота, но принципът е един: стойностите на блокове и иноди могат да се ограничат. *Само променете стойностите на soft и hard*. Ако не е посочено друго, блоковете за по 1k. Периодът на благосклонност се задава с `edquota -t`. Например:

```
# edquota -u colin
```

Linux

```
Дискови квоти за потребител colin (uid 1007):
  Filesystem         blocks       soft       hard     inodes     soft     hard
  /dev/sda8             108       1000       2000          1        0        0
```

FreeBSD

```
Квоти за потребител colin:
/home: kbytes in use: 504184, limits (soft = 700000, hard = 800000)
    inodes in use: 1792, limits (soft = 0, hard = 0)
```

За много потребители

Командата `edquota -p` се използва за дуплициране на квотата за други потребители. Например за дуплициране на примерна квота за всички потребители:

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

Проверки

Потребителите могат за проверяват квотата си със `quota` (файлът quota.user трябва да е четим). Root може да проверява всички квоти.

```
# quota -u colin                       # Проверка квотата за потребител
# repquota /home                       # Пълен отчет за дял за всички потребители
```

Резервиране и възстановяване на всички бази данни:

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

Тук "secret" е mysql root паролата, няма шпация след -p. Когато опцията -p е използвана (с/без парола), паролата се изисква в командния промпт.

18.3 SQLite

SQLite[13] е малка и мощна SQL база данни, която е: самосъдържаща се, работеща без сървър и не изискваща конфигурация.

Копиране и възстановяване

Може да е полезно да копирате и да възстановите SQLite база данни. Например можете да промените копирания файл, за да промените атрибут или тип от колона и после да възстановите базата данни. Тов ае по-лесно от изпращане на SQL командите. Използвайте командата `sqlite3` за 3.x база данни.

```
# sqlite database.db .dump > dump.sql              # копиране
# sqlite database.db < dump.sql                    # възстановяване
```

Конвертиране от 2.x в 3.x база данни

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

19 Дискова квота

Дисковата квота позволява ограничение на дисковото място и/или броя файлове потребител (или член на група) може да ползва. Квотите се задават на пофайлов принцип и се задействат в ядрото.

19.1 Linux настройка

Пакета за инструменти за квота обикновено трябва да бъде инсталиран, той съдържа инструменти на команден ред.
Активирайте потребителска квота в fstab и премонтирайте дяла. Ако дяла е зает или заключените файлове трябва да се затворят, или да се рестартира системата. Добавете `usrquota` към fstab опциите за монтиране, например:

```
/dev/sda2     /home    reiserfs     rw,acl,user_xattr,usrquota 1 1
# mount -o remount /home
# mount                              # Проверка за активност на квота, иначе рестарт
```

Инициализиране на quota.user файла със `quotacheck`.

```
# quotacheck -vum /home
# chmod 644 /home/aquota.user        # За позволяване на потребителите да проверяват собствената
```

Активиране на квота със добавения скрипт (т.е. /etc/init.d/quotad в SuSE) или с `quotaon`:

```
quotaon -vu /home
```

Проверка за активност на квота:

```
quota -v
```

13.http://www.sqlite.org

Декларирано в /etc/fstab:

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

Прикачване на FreeBSD дял с Linux

Намерете номера на дяла съдържан във fdisk, това обикновено е root дяла, но може също да бъде и на друг BSD дял. Ако FreeBSD има много дялове, това са онези недекларирани във fdisk таблицата, но видими в /dev/sda* или /dev/hda*.

```
# fdisk /dev/sda                     # Намира FreeBSD дяла
/dev/sda3   *        5357        7905    20474842+  a5  FreeBSD
# mount -t ufs -o ufstype=ufs2,ro /dev/sda3 /mnt
/dev/sda10 = /tmp; /dev/sda11 /usr   # Другите дялове
```

Преприкачане

Преприкачване на устройство без да бъде откачвано. необходимо за `fsck` например:

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

Копиране на сурови (неформатирани/необработени) данни от компакт диск в iso дисков образ:

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

3.7 Прикачване на SMB споделена част

Да предположим, че искаме достъп до споделен SMB дял myshare на компютър или smb-сървър, адресът, както е въведен до Windows компютъра е \\smbserver\myshare\. Ние прикачваме в /mnt/smbshare. Забележка> cifs изисква IP или DNS име, не Windows име.

Linux

```
# smbclient -U user -I 192.168.16.229 -L //smbshare/    # Показва споделените дялове
# 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
```

Допълнително с пакета mount.cifs е възможно да запишем акредитивите във файл, на пример `/home/user/.smb`:

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

Добавяне на прикачване както следва:

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

FreeBSD

Използвайте -I, за да зададете IP (или DNS име); smbserver е Windows името.

```
# smbutil view -I 192.168.16.229 //winuser@smbserver    # Показва споделените дялове
# mount_smbfs -I 192.168.16.229 //winuser@smbserver/myshare /mnt/smbshare
```

3.8 Прикачване на дисков образ

Linux loop-back

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

FreeBSD

С памет-устройство (изпълнете # kldload md.ko ако е необходимо):

```
# mdconfig -a -t vnode -f file.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
# umount /mnt; mdconfig -d -u 0                          # Изчиства md устройството
```

Или с виртуален възел:

```
# vnconfig /dev/vn0c file.iso; mount -t cd9660 /dev/vn0c /mnt
# umount /mnt; vnconfig -u /dev/vn0c                     # Изчиства vn устройството
```

Solaris и FreeBSD

с loop-back файлов интерфейс или lofi:

```
# lofiadm -a file.iso
# mount -F hsfs -o ro /dev/lofi/1 /mnt
# umount /mnt; lofiadm -d /dev/lofi/1                    # Изчиства lofi устройство
```

3.9 Създаване и записване на ISO образ

Това ще копира компакт диск или DVD сектор по сектор. Без `conv=notrunc`, образът ще бъде по-малък ако има по-малко съдържание на диска. Виж по-долу и dd примерите (page 39).

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

Използвайте mkisofs за създаване на CD/DVD образ от файлове в директория. За да преодолеете ограниченията за имена на файловете: -r разрешава Rock Ridge разширения обичайни за UNIX системи, -J разрешава Joliet разширения използвани от Майкрософтски системи. -L разрешава ISO9660 имена на файловете да започват с период.

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

На FreeBSD, mkisofs е открит в портовете в sysutils/cdrtools.

Записване на CD/DVD ISO образ

FreeBSD

FreeBSD не разрешава DMA на ATAPI устройства по подразбиране. DMA е разрешен с командата sysctl и аргументите по-долу, или с /boot/loader.conf със следните въвеждания:

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

Използвайте `burncd` с ATAPI устройство (`burncd` е част от основната система) и `cdrecord` (в sysutils/cdrtools) със SCSI устройство.

```
# burncd -f /dev/acd0 data imagefile.iso fixate      # За ATAPI устройство
# cdrecord -scanbus                 # За откриване на записващото устройство (като 1,0,0)
# cdrecord dev=1,0,0 imagefile.iso
```

Linux

Също използвайте `cdrecord` на Linux както е описано по-горе. Допълнително е възможно да използвате чистия ATAPI интерфейс, който се намира с:

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

И записване на CD/DVD както по-горе.

Конвертиране на Nero .nrg файл в .iso

Nero просто добавя 300Kb header към нормалния iso образ. Това може да бъде уредено с dd.

Резерва и възстановяване

Резервирането и възстановяването се правят с потребителя pgsql или postgres. Резервиране и възстановяване на една база данни:

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

Резервиране и възстановяване на всички бази данни (включително потребители):

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

В този случай възстановяването е стартирано с базата данни postgres, което е по-добре когато се презарежда празна група.

18.2 MySQL

Промяна на mysql root или потребителска парола

Метод 1

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

Метод 2

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

Създаване на потребител и база данни

```
# mysql -u root mysql
mysql> CREATE DATABASE bobdb;
mysql> GRANT ALL ON *.* TO 'bob'@'%' IDENTIFIED BY 'pwd'; # Използвайте localhost вместо %
                                                          # за ограничаване на достъпа до мрежата
mysql> DROP DATABASE bobdb;                               # Изтрива база данни
mysql> DROP USER bob;                                     # Изтрива потребител
mysql> DELETE FROM mysql.user WHERE user='bob and host='hostname'; # Алт. команда
mysql> FLUSH PRIVILEGES;
```

Предоставяне на отделечен достъп

Отдалечения достъп обикновено е разрешен за една база данни, а не за всичките. Файлът `/etc/my.cnf` съдържа IP адреса, към който да се свърже. Обикновено трябва да махнете коментара от реда `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;                  # Използвайте 'hostname' или също '%' за пълен достъп
```

Резерва и възстановяване

Резервиране и възстановяване на една база данни:

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

16.5 Копиране на аудио компакт диск

Програмата `cdparanoia`[12] може да запише аудио парчета (FreeBSD порт в audio/cdparanoia/), `oggenc` може да кодира в Ogg Vorbis формат, `lame` конвертира в mp3.

```
# cdparanoia -B                       # Кпира парчетата в wav файлове в настоящата директория
# lame -b 256 in.wav out.mp3          # Кодира в mp3 256 kb/s
# for i in *.wav; do lame -b 256 $i `basename $i .wav`.mp3; done
# oggenc in.wav -b 256 out.ogg        # Кодира в Ogg Vorbis 256 kb/s
```

17 Принтиране

17.1 Принтиране с lpr

```
# lpr unixtoolbox.ps                  # Принтира на принтера по подразбиране
# export PRINTER=hp4600               # Променя принтера по подразбиране
# lpr -Php4500 #2 unixtoolbox.ps      # Използва принтер hp4500 и принтира 2 копия
# lpr -o Duplex=DuplexNoTumble ...    # Принтира дуплекс по дългата страна
# lpr -o PageSize=A4,Duplex=DuplexNoTumble ...
# lpq                                 # Проверява опашката на принтера по подразбиране
# lpq -l -Php4500                     # Опашката на принтер hp4500 с многословие
# lprm -                              # Премахва всички потребителски задачи от принтера по подраз
# lprm -Php4500 3186                  # Премахва задача 3186. Намирате номера на задача с lpq
# lpc status                          # Списък със всички налични принтери
# lpc status hp4500                   # Проверява дали принтера е на линия и дължината на опашката
```

18 Бази данни

18.1 PostgreSQL

Промяна на root или потребителска парола

```
# psql -d template1 -U pgsql
> alter user pgsql with password 'pgsql_password';  # Използвайте потребислко име вместо "pgsql"
```

Създаване на потребител и база данни

Командите `createuser`, `dropuser`, `createdb` и `dropdb` са удобни преки пътища еквиваленти на SQL командите. Новият потребител е bob с база данни bobdb ; използване като root с pgsql супер потребителя на базата данни:

```
# createuser -U pgsql -P bob          # -P ще попита за парола
# createdb -U pgsql -O bob bobdb      # новата bobdb принадлежи на bob
# dropdb bobdb                        # Изтрива базата данни bobdb
# dropuser bob                        # Изтрива потребителя bob
```

Главния механизъм за автентикация на базата данни е конфигуриран в pg_hba.conf

Предоставяне на отдалечен достъп

Файлът `$PGSQL_DATA_D/postgresql.conf` оказва адреса, към който да се свърже. Обикновено `listen_addresses = '*'` за Postgres 8.x.
Файлът `$PGSQL_DATA_D/pg_hba.conf` дефинира контролра върху достъпа. Примери:

```
# 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
```

12.http://xiph.org/paranoia/


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

Конвертиране на bin/cue образ в .iso

Малката `bchunk` програма[1] може да направи това. Тя е във FreeBSD портовете в sysutils/ bchunk.

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

3.10 Създаване ан файлов-базиран образ

На пример дял от 1GB използвайки файла /usr/vdisk.img.

FreeBSD

```
# dd if=/dev/random of=/usr/vdisk.img bs=1K count=1M
# mdconfig -a -t vnode -f /usr/vdisk.img -u 1         # Създава устройство /dev/md1
# bsdlabel -w /dev/md1
# newfs /dev/md1c
# mount /dev/md1c /mnt
# umount /mnt; mdconfig -d -u 1; rm /usr/vdisk.img    # Почиства md устройството
```

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                      # Почистване
```

Linux с losetup

`/dev/zero` е много по-бърз от `urandom`, но по-малко сигурен за криптиране.

```
# dd if=/dev/urandom of=/usr/vdisk.img bs=1024k count=1024
# losetup /dev/loop0 /usr/vdisk.img                   # Създава и асоциира /dev/loop0
# mkfs.ext3 /dev/loop0
# mount /dev/loop0 /mnt
# losetup -a                                          # Проверява използваните серии (loops)
# umount /mnt
# losetup -d /dev/loop0                               # Отделяне
# rm /usr/vdisk.img
```

3.11 Създаване на memory файлова система

Memory-базирана файлова система е много бърза за тежки IO програми. Как да създадете 64 MB дял прикачен към /memdisk:

FreeBSD

```
# mount_mfs -o rw -s 64M md /memdisk
# umount /memdisk; mdconfig -d -u 0                   # Почиства md устройството
md      /memdisk      mfs      rw,-s64M    0   0     # /etc/fstab въвеждане
```

Linux

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

1.http://freshmeat.net/projects/bchunk/

3.12 Диск производителностDisk performance

Четене и записване на 1 GB файл на дял 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      # Само за Linux
```

4 Мрежа

Рутиране (p14) | Допълнително IP (p15) | Промяна на MAC (p15) | Портове (p15) | Firewall (p15) | IP Препращане (p16) | NAT (p16) | DNS (p16) | DHCP (p18) | Трафик (p18) | QoS (p19) | NIS (p20)

4.1 Откриване на проблеми(Виж също Анализ на трафика) (page 18)

```
# mii-diag eth0             # Покажи статуса на връзката(Linux)
# ifconfig fxp0             # Проверка на "media" полето в FreeBSD
# arp -a                    # Проверка на рутер (или хост) ARP записа (вс. OS)
# ping cb.vu                # Първото нещо дето пробваме...
# traceroute cb.vu          # Покажи пътя на рутиране до дестинацията
# mii-diag -F 100baseTx-FD eth0  # Форсиране на 100Mbit Full duplex (Linux)
# ifconfig fxp0 media 100baseTX mediaopt full-duplex  # Същото за FreeBSD
# netstat -s                # Системна статистика за всички мрежови протоколи
```

4.2 Рутиране

Показване на рутинг таблицата

```
# route -n                  # Linux
# netstat -rn               # Linux, BSD и UNIX
# route print               # Windows
```

Добавяне и изтриване на рутиране

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
```

Добавяне на перманентно рутиране 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       # същото с ip рутиране
# 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               # същото с ip рутиране
# route delete -net 192.168.20.0 netmask 255.255.255.0
```

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
```

Използвай add -p за перманентно рутиране.


```
# ldd /usr/bin/rsync                  # списък с всички необходими библиотеки за работа
# ldconfig -n /path/to/libs/          # Добавя път към зависимостите за споделените споделените би
# ldconfig -m /path/to/libs/          # FreeBSD
# LD_LIBRARY_PATH                     # Променливата слага връзка към пътя до библиотеката (пряк п
```

16 Конвертиране на медия

Понякога някой просто се нуждае да конвертира видео, аудио файл или документ в друг формат.

16.1 Текст кодиране

Текст кодирането може да стане напълно грешно, особено когато езика изисква специални символи като àäç. ТКомандата `iconv` може да конвертира от една кодировка в друга.

```
# iconv -f <from_encoding> -t <to_encoding> <input_file>
# iconv -f ISO8859-1 -t UTF-8 -o file.input > file_utf8
# iconv -l                            # Списък с познатите набори от кодирани символи
```

С опцията -f, iconv ще използва локалния набор от символи, което обикновено е добре ако документа се показва добре.

16.2 Unix - DOS нови редове

Конвертиране от DOS (CR/LF) към Unix (LF) нови редове в Unix шел. Виж също `dos2unix` и `unix2dos` ако ги имате.

```
# sed 's/.$//' dosfile.txt > unixfile.txt
```

Конвертиране от Unix към DOS нови редове в Windows среда. Използвай sed от mingw или cygwin.

```
# sed -n p unixfile.txt > dosfile.txt
```

16.3 PDF към Jpeg и съединяване на PDF файлове

Конвертиране на PDF документ с `gs` (GhostScript) в jpeg (или png) изображения за всяка страница. Също така много по-кратко с `convert` (от ImageMagick или GraphicsMagick).

```
# gs -dBATCH -dNOPAUSE -sDEVICE=jpeg -r150 -dTextAlphaBits=4 -dGraphicsAlphaBits=4 \
 -dMaxStripSize=8192 -sOutputFile=unixtoolbox_%d.jpg unixtoolbox.pdf
# convert unixtoolbox.pdf unixtoolbox-%03d.png
# convert *.jpeg images.pdf           # Създава прост PDF с всички изображения
```

Ghostscript може също и да обедини няколко pdf файла в един.

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

16.4 Конвертиране на видео

Компресиране на Canon digicam видео с mpeg4 кодек и поправяне на повредения звук.

```
# 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
```

```
# pwd                                 # Показва работната директория
# mkdir -p /path/to/dir               # без съобщение за грешка ако съществува, създава и съответн
# rmdir /path/to/dir                  # Премахва празна директория
# rm -rf /path/to/dir                 # Премахва директория и нейното съдържание (принудително)
# cp -la /dir1 /dir2                  # Архивира и свързва твърдо файлове вместо да ги копира
# cp -lpR /dir1 /dir2                 # Същото за FreeBSD
# mv /dir1 /dir2                      # Преименува директория
```

15 Инсталиране на софтуер

15.1 Списък с инсталираните пакети

```
# rpm -qa                             # Списък с инсталираните пакети (RH, SuSE, RPM базирани)
# dpkg -l                             # Debian, Ubuntu
# pkg_info                            # FreeBSD списък с всички инсталирани пакети
# pkg_info -W smbd                    # FreeBSD показва към кой пакет принадлежи smbd
# pkginfo                             # Solaris
```

15.2 Добавяне/премахване на софтуер

Обвивки: yast2/yast за SuSE, redhat-config-packages за Red Hat.

```
# rpm -i пакет_име.rpm                # инсталира пакета (RH, SuSE, RPM базирани)
# rpm -e пакет_име                    # Премахва пакет
```

Debian

```
# apt-get update                      # Първо обновява списъка с пакети
# apt-get install emacs               # Инсталира пакета emacs
# dpkg --remove emacs                 # Премахва пакета emacs
```

FreeBSD

```
# pkg_add -r rsync                    # Взима и инсталира rsync.
# pkg_delete /var/db/pkg/rsync-xx     # Изтрива пакета rsync
```

Задайте от къде да се взимат пакетите чрез `PACKAGESITE` променливата. На пример:

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

FreeBSD портове

Дървото на портовете `/usr/ports/` е колекция от софтуер, готов за компилиране и инсталиране. Портовете се обновяват с програмата `portsnap`.

```
# portsnap fetch extract              # Създава дървото при стартиране за първи път
# portsnap fetch update               # Обновява дървото с портовете
# cd /usr/ports/net/rsync/            # Избира пакет за инсталиране
# make install distclean              # Инслира и почиства (всиж също man ports)
# make package                        # Създава изпълним пакет за порта
```

15.3 Път до библиотеката

Поради комплекси зависимости и свързане по време на работа (runtime), програмите са трудни за копиране на друга система или дистрибуция. Все пак за малки програми с малко зависимости, либсващите библиотеки могат да бъдат прехвърлени. Библиотеките за работно време (и липсващите) се проверяват с `ldd` и контролират с `ldconfig`.

4.3 Конфигурация на втори IP адрес

Linux

```
# ifconfig eth0 192.168.50.254 netmask 255.255.255.0       # Първи IP
# ifconfig eth0:0 192.168.51.254 netmask 255.255.255.0     # Втори IP
```

FreeBSD

```
# ifconfig fxp0 inet 192.168.50.254/24                     # Първи IP
# ifconfig fxp0 alias 192.168.51.254 netmask 255.255.255.0 # Втори IP
```

Перманентно добавяне 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"
```

4.4 Промяна на MAC адрес

```
# ifconfig eth0 hw ether 00:01:02:03:04:05      # Linux
# ifconfig fxp0 link 00:01:02:03:04:05          # FreeBSD
```

4.5 Използвани портове

Отворени портове за слушане:

```
# netstat -an | grep LISTEN
# lsof -i                    # Linux списък на всички интернет връзки
# socklist                   # Linux списък на всички отворени гнезда/сокети
# sockstat -4                # FreeBSD списък на приложенията
# netstat -anp --udp --tcp | grep LISTEN        # Linux
# netstat -tup               # Списък на активните връзки от/към системата (Linux)
# netstat -tupl              # Списък на слушащите портове (Linux)
# netstat -ano               # Windows
```

4.6 Защитна стена

Проверка за работеща защитна стена (само за типична конфигурация):

Linux

```
# iptables -L -n -v                   # Статус
Отваране на iptables защитна стена
# iptables -Z                         # Нулиране на пакетите и байтовите броячи във всички правила
# iptables -F                         # Нулиране на всички правила
# iptables -X                         # Изтриване на всички правила
# iptables -P INPUT       ACCEPT      # Отваряне на всичко
# iptables -P FORWARD     ACCEPT
# iptables -P OUTPUT      ACCEPT
```

FreeBSD

```
# ipfw show                           # Статус
# ipfw list 65535                     # ако отговора е "65535 deny ip from any to any" защитната с
# sysctl net.inet.ip.fw.enable=0      # Изключване
# sysctl net.inet.ip.fw.enable=1      # Включване
```

4.7 IP Препращане за рутиране

Linux

Проверка и включване на IP препращане с:

```
# cat /proc/sys/net/ipv4/ip_forward  # Проверка на IP препращане 0=изключено, 1=включено
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

or edit /etc/sysctl.conf with:

```
net.ipv4.ip_forward = 1
```

FreeBSD

Проверка и включване с:

```
# sysctl net.inet.ip.forwarding      # Проверка на IP препращане 0=изключено, 1=включено
# sysctl net.inet.ip.forwarding=1
# sysctl net.inet.ip.fastforwarding=1        # За посветен рутер или защитна стена
Перманентно добавяне в /etc/rc.conf:
gateway_enable="YES"                 # Поставяне на YES ако хоста ще е gateway.
```

4.8 NAT - Транслация на мрежови адреси

Linux

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE        # за активиране на NAT
# iptables -t nat -A PREROUTING -p tcp -d 78.31.70.238 --dport 20022 -j DNAT \
--to 192.168.16.44:22           # Препращане на порт 20022 към вътрешен IP порт на 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     # Препращане на портове в обхвата 993-995
# ip route flush cache
# iptables -L -t nat            # Проверка на NAT статуса
```

Изтриване на препращането с -D вместо -A.

FreeBSD

```
# natd -s -m -u -dynamic -f /etc/natd.conf -n fxp0
Or edit /etc/rc.conf with:
firewall_enable="YES"           # Поставяме YES за включване на защитната стена
firewall_type="open"            # Тип защитна стена (виж /etc/rc.firewall)
natd_enable="YES"               # Включване на natd (ако firewall_enable == YES).
natd_interface="tun0"           # Публичен интерфейс или IP адрес за използване.
natd_flags="-s -m -u -dynamic -f /etc/natd.conf"
```

Препращане на портове:

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

4.9 DNS

В Unix DNS записите са валидни за всички интерфейси и се описват в /etc/resolv.conf. Домейнът, в който хостът трябва да се намира също се описва в този файл. Минималната конфигурация е:

Screen команди (в screen)

Всички screen команди започват с **Ctrl-a**.

- **Ctrl-a ?** помощ и обобщение на функциите
- **Ctrl-a c** създава нов прозорец (терминал)
- **Ctrl-a Ctrl-n and Ctrl-a Ctrl-p** превключва към следващия или предишния прозоред в списъка, по номер.
- **Ctrl-a Ctrl-N** където N е число от 0 до 9, за превключване към съответстващия прозорец.
- **Ctrl-a "** за получаване на управляем списък с работещите прозорци
- **Ctrl-a a** за изчистване на пропуснат Ctrl-a
- **Ctrl-a Ctrl-d** за прекъсване и напускане на сесията работеща във фона
- **Ctrl-a x** заключва screen терминала с парола

Screen сесията е прекъсната, когато програмата в работещия терминал е затворена и вие излезете от терминала.

14.7 Find

Някои важни опции:

`-x` (на BSD) `-xdev` (на Linux) Остава на същата файлова система (dev в fstab).
`-exec cmd {} \;` Изпълнява команда и заменя {} с пълня път
`-iname` Като -name, но се влияе от случая
`-ls` Показва информация за файла (като ls -la)
`-size n` n e +-n (k M G T P)
`-cmin n` Статуса на файла беше променен преди n минути.

```
# find . -type f ! -perm -444         # Намира файлове, които немогат да бъдат четени от всеки.
# find . -type d ! -perm -111         # Намира директории, които не са достъпни за всички.
# find /home/user/ -cmin 10 -print    # Файлове създадени или променени в последните 10 минути.
# find . -name '*.[ch]' | xargs grep -E 'expr' # Търси 'expr' в тази директория и следващите под
# find / -name "*.core" | xargs rm    # Намира core dumps[1] и ги изтрива
# find / -name "*.core" -print -exec rm {} \;  # Other syntax
# find . \( -name "*.png" -o -name "*.jpg" \) -print
                                      # iname is not case sensitive
# find . \( -iname "*.png" -o -iname "*.jpg" \) -print -exec tar -rf images.tar {} \;
# find . -type f -name "*.txt" ! -name README.txt -print  # Exclude README.txt files
# find /var/ -size +1M -exec ls -lh {} \;
# find /var/ -size +1M -ls            # This is simpler
# find . -size +10M -size -50M -print
# find /usr/ports/ -name work -type d -print -exec rm -rf {} \;  # Clean the ports
    Find files with SUID; those file have to be kept secure
# find / -type f -user root -perm -4000 -exec ls -l {} \;
```

[1] core dump - е снимка на прекъснат процес записан при дебъгване. Основния файл е създаден под името **core** в настоящата директория на процеса, когато някое нетипично събитие е причината за прекъсването му.

14.8 Разни

```
# which command                       # Показва пълния път за команда
# time command                        # Показва колко време е необходимо за изпълление
# time cat                            # Използва time като хронометър. Ctrl-c за прекъсване
# set | grep $USER                    # Списък с настоящата среда
# cal -3                              # Показва тримесечен календар
# date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
# date 10022155                       # Задава дата и час
# whatis grep                         # Показва кратка информация за команда или дума
# whereis java                        # Търси път и стандартни директории за дума
# setenv varname value                # Задава променлива за средата varname към value (csh/tcsh)
# export varname="value"              # Задава променлива за средата varname към value (sh/ksh/bas
```

```
# dd bs=1M if=/dev/ad4s3e | gzip | ssh eedcoba@fry 'dd of=ad4s3e.gz' # същото отдалечено
# 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    # Пропуска MBR
    # Това е необходимо ако дестинацията (ad2) е по-малка.
```

Възстановяване

Командата `dd` ще прочете *всеки отделен блок* от дяла, дори блоковете. В случай на проблеми по-добре е да използвате опцията `conv=sync,noerror` така dd ще пропусне лоши сектори и ще запише нули в дестинацията. Съответно е важно да сложите размера на блока на равен или по-малък от размера на дисковия блок. Размер от 1k изглежда безопасен, сложете го с `bs=1k`. Ако диска има лоши сектори и данните трябва да бъдат възстановени от дял, създайте файл-снимка с dd, прикачете снимката и копирайте съдържанието на нов диск. С опцията `noerror`, dd ще пропусне лошите сектори и вместо това ще запише нули, от това само данните съдържащи се в лошите сектори ще бъдат загубени.

```
# dd if=/dev/hda of=/dev/null bs=1m                  # Проверява за лоши блокове
# dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc | gzip | ssh \ # Изпраща до отдалечен
root@fry 'dd of=hda1.gz bs=1k'
# dd bs=1k if=/dev/hda1 conv=sync,noerror,notrunc of=hda1.img    # Съхранява във файл-снимка
# mount -o loop /hda1.img /mnt                       # Прикача снимката (page 13)
# rsync -ax /mnt/ /newdisk/                          # Копира на нов диск
# dd if=/dev/hda of=/dev/hda                         # Опреснява магнетичното състояние
  # Горното е полезно за обновяване на диск. Напълно е безопасно, но трябва да бъде откачен (unmc
```

Изтриване

```
# dd if=/dev/zero of=/dev/hdc count=1                # Изтрива MBR и таблицата на дяла
# dd if=/dev/zero of=/dev/hdc                        # Изтрива целия диск
# dd if=/dev/urandom of=/dev/hdc                     # Изтрива целия диск по-добре
# kill -USR1 PID                                     # Показва прогреса на dd (Само за Linux!)
```

14.6 screen

Screen има две основни функции:

- Стартиране на няколко конзолни сесии в една конзола/терминал.
- Стартирана програма е отделена от истинския терминал и така може да работи на фона. Истинския терминал може да бъде затворен и прикачен по-късно.

Кратък пример за стартиране

стартирайте screen с:

```
# screen
```

В screen сесията можем да стартираме дълго работеща програма (отгоре на останалите). Откачаме терминала и прикачаме същия терминал от друга машина (през ssh например).

```
# top
```

Сега разкачаме с **Ctrl-a Ctrl-d**. Прикачаме терминала с

```
# screen -r
```

или по-добре:

```
# screen -R -D
```

Прикача тук и сега. По-подробно това означава: Ако сесията работи, тогава преприкачи. Ако е необходимо първо разкачете и излезте отдалечено. Ако не е стартирана я създайте и осведомете потребителя.


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

Проверка на името на домейна с:

```
# hostname -d                          # Същото като dnsdomainname
```

Windows

Във Windows DNS се конфигурира за интерфейс. За показване конфигурацията на DNS и изтриване на DNS кеша:

```
# ipconfig /?                          # Помощ
# ipconfig /all                        # Цялата информация вкл. DNS
# ipconfig /flushdns                   # Изтриване на DNS кеша
```

Препращане на заявки

dig е вашия инструмент за проверка на DNS настройките. Например публичен DNS сървър `213.133.105.2 ns.second-ns.de` може да се използва за тест. Вижте от кой сървър клиента получава отговор (упростен отговор).

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

Рутерът 192.168.51.254 отговоря и отговорът е A запис. Всеки запис може да бъде извикан и DNS сървърът може да се избере с @:

```
# dig MX google.com
# dig @127.0.0.1 NS sun.com            # За тест на локалния сървър
# dig @204.97.212.10 NS MX heise.de    # Заявка към външен сървър
# dig AXFR @ns1.xname.org cb.vu        # Получаване на цялата зона(трансфер на зона)
```

Програмата host също е мощна.

```
# host -t MX cb.vu                     # Получаване на пощенските MX записи
# host -t NS -T sun.com                # получаване на NS запис през TCP връзка
# host -a sleepyowl.net                # Получаване на всичко
```

Обратни заявки

Намиране на име по IP адрес (in-addr.arpa.). Може да бъде направено с `dig`, `host` и `nslookup`:

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

/etc/hosts

Единични хостове могат да се конфигурират в /etc/hosts вместо стартиране на `named` локално за разрешаване на заявките по имена. Форматът е прост, например:

```
78.31.70.238   sleepyowl.net   sleepyowl
```

Приоритетът между hosts и dns заявка, т.е. редът на резолюцията по име, може да се конфигурира в `/etc/nsswitch.conf` * И * /etc/host.conf. Файлът съществува и във Windows, обикновено в:

```
C:\WINDOWS\SYSTEM32\DRIVERS\ETC
```

4.10 DHCP

Linux

Някой дистрибуции (SuSE) използват dhcpcd като клиент. Интерфайса по подразбиране е eth0.

```
# dhcpcd -n eth0           # Предизвиква обновяване
# dhcpcd -k eth0           # освобождаване и изключване
```

Обхвата с пълната информация се намира в:

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

FreeBSD

FreeBSD (и Debian) използват dhclient. За конфигурация на интерфейс (например bge0) стартирайте:

```
# dhclient bge0
```

Обхвата с пълната информация се намира в:

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

Използвайте

```
/etc/dhclient.conf
```

за настройка на опциите или поставяне на други опции:

```
# 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 обхвата може да бъде обновен с `ipconfig`:

```
# ipconfig /renew          # обновяване на всички адаптери
# ipconfig /renew LAN      # обновяване на адаптера "LAN"
# ipconfig /release WLAN   # освобождаване на адаптера "WLAN"
```

Да, добра идея е да си именувате адаптера с просто име!

4.11 Анализ на трафика

Bmon[2] е малък конзолен монитор на лентата и може да показва потока през различните интерфейси.

Сниф/подслушване с tcpdump

```
# tcpdump -nl -i bge0 not port ssh and src \(192.168.16.121 or 192.168.16.54\)
# tcpdump -l > dump && tail -f dump              # Буфериран изход
# tcpdump -i rl0 -w traffic.rl0                  # Записване на трафика в бинарен файл
# tcpdump -r traffic.rl0                         # Четене от файл (също за ethereal
# tcpdump port 80                                # Две класически команди
# tcpdump host google.com
# tcpdump -i eth0 -X port \(110 or 143\)         # Проверка дали pop или imap са сигурни
# tcpdump -n -i eth0 icmp                        # Записване само на ping
# tcpdump -i eth0 -s 0 -A port 80 | grep GET     # -s 0 за пълен пакет -A за ASCII
```

Още важни опции:

2.http://people.suug.ch/~tgr/bmon/

14.4 tar

Командата `tar` (tape архив) създава и извлича авхиви от файлове и директории. Архивът .tar не е компресиран, компресиран архив има разширението .tgz или .tar.gz (zip) или .tbz (bzip2). Не използвайте пълен път когато създавате архив, вероятно искате да го разархивирате в другаде. Някои полезни команди са:

Създаване

```
# cd /
# tar -cf home.tar home/        # архивира цялата /home директория (c за създаване)
# tar -czf home.tgz home/       # същото със zip компресия
# tar -cjf home.tbz home/       # същото с bzip2 компресия
```

Само вмъква една (или две) директории от дърво, но запазва относителната структура. За примерен архиве /usr/local/etc и /usr/local/www и първата директория в архива трябва да бъде local/.

```
# tar -C /usr -czf local.tgz local/etc local/www
# tar -C /usr -xzf local.tgz    # За разархивиране локалната директория в /usr
# cd /usr; tar -xzf local.tgz   # е същото като горното
```

Разархивиране

```
# tar -tzf home.tgz             # поглежда в архива без да го разархивира (списък)
# tar -xf home.tar              # разархивира архива тук (x за разархивиране)
# tar -xzf home.tgz             # същото със zip компресия
# tar -xjf home.tgz             # същото с bzip2 компресия
# tar -xjf home.tgz home/colin/file.txt    # възстановява отделен файл
```

По-разширено

```
# tar c dir/ | gzip | ssh user@remote 'dd of=dir.tgz' # архивира dir/ и записва отдалечено.
# tar cvf - `find . -print` > backup.tar              # архивира настоящата директория.
# tar -cf - -C /etc . | tar xpf - -C /backup/etc     # Копира директории
# tar -cf - -C /etc . | ssh user@remote tar xpf - -C /backup/etc      # Отдалечено копиране.
# tar -czf home.tgz --exclude '*.o' --exclude 'tmp/' home/
```

14.5 dd

Програмата `dd` (disk dump) се използва за копиране на дялове и дискове и за други трикове с копиране. Типично използване:

```
# dd if=<source> of=<target> bs=<byte size> conv=<conversion>
```

Важни опции при конвертиране:

- `notrunc` не отрязва крайния файл, всички нули ще бъдат записани като нули.
- `noerror` продължава след прочитане на грашки (н.пр. лоши блокове)
- `sync` попълва всеки входящ блок с нули до ibs-размер

По подразбиране байт-размера е 512 (един блок). MBR, където таблицата на дяла се намира, е във първия блок, Първите 63 блока от диска са празни. По-големи байт-размери се копират по-бързо, но изискват повече памет.

Резерва и възвръщане

```
# dd if=/dev/hda of=/dev/hdc bs=16065b                # копира диск в диск (със същия размер)
# dd if=/dev/sda7 of /home/root.img bs=4096 conv=notrunc,noerror # резерва /
# dd if /home/root.img of=/dev/sda7 bs=4096 conv=notrunc,noerror # възвръщане /
# dd bs=1M if=/dev/ad4s3e | gzip -c > ad4s3e.gz                 # архивиране чрез Zip на резерва
# gunzip -dc ad4s3e.gz | dd of=/dev/ad0s3e bs=1M                # възстановяване на zip
```

?pattern Търси назад за (N-тия) съответстващ ред.
n Повтори предишното търсене (за N-тото срещане).
N Повтаря последното търсене в обратна посока.
q изход

14.2 vi

Vi е наличен във ВСЯКА Linux/Unix инсталация и затова е полезно да знаете някои основни команди. Има два режима: команден режим и въвеждащ режим. Командния режим се включва с **[ESC]**, въвеждащият режим с **i**.

Изход

:w новоименафайла записва файла в новоименафайла
:wq или :x запис и изход
:q! изход без запис

Търсене и местене

/стринг Търси напред за стринг
?стринг Търси назад за стринг
n Търси за следващото появяване на стринг
N Търси за предишното появяване на стринг
{ Премества с параграф назад
} Премества с параграф напред
1G Премества до първия ред на файла
nG Премества до n-тия ред на файла
G Премества до последния ред на файла
:%s/OLD/NEW/g Търси и замества всеки резултат

Изтриване на текст

dd Изтрива настоящия ред
D Изтрива до края на реда
dw Изтрива дума
x Изтрива символ
u Връща последното
U Връща всички промени за съответния ред

14.3 mail

Командата `mail` е просто приложение за четене и пращане на мейл, обикновено е инсталирано. за да изпратите мейл просто напишете "mail user@domain". Първия ред е заглавието, след това съдържанието на писмото. Прекъсвнете и изпратете писмото с една точка (.) на нов ред. На 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
#
```

Това също работи и с лула:

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

This is also a simple way to test the mail server.

`-A` Показва всеки пакет в текстов вид (без "шапка")
`-X` Показва пакетите в hex и ASCII
`-l` Буферира стандартния изход
`-D` Показва достъпните интерфейси

Във Windows използвайте windump от www.winpcap.org. Използвайте windump -D за списък на интерфейсите.

Сканиране с nmap

Nmap[3] е порт скенер с OS детекция, обикновено е инсталиран в повечето дистрибуции и е достъпен и за Windows. Ако вие не сканирате вашите сървъри, хакерите го правят за вас...

```
# nmap cb.vu                # сканира всички резервирани TCP портове на хоста
# nmap -sP 192.168.16.0/24 # Показва кой IP е използван от кой хост в 0/24
# nmap -sS -sV -O cb.vu     # Прави невидимо SYN сканиране със версия и OS детекция
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)
```

4.12 Контрол на трафика (QoS)

Контрола на трафика управлява заявките, политиките, разпределението и други параметри на трафика в мрежа. Следващите примери са за прости практически приложения на Linux и FreeBSD възможностите за по-добро използване на достъпната лента.

Ограничаване на ъплоуда

DSL или кабелните модеми имат дълга опашка за подобрение на производителността на ъплоуд. Обаче запълването на опашката от бързо устройство (мрежова карта) драстично ще намали интерактивността. Следователно е полезно да се намали нивото на ъплоуд на устройството, за да се напасне с това на модема, това би подобрило много интерактивността. Настройте на около 90% от максималната модемна (кабелна) скорост.

Linux

За 512 Kbit ъплоуд модем.

```
# 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                       # Изтриване на опашката
# tc qdisc change dev eth0 root tbf rate 220kbit latency 50ms burst 1540
```

FreeBSD

FreeBSD използва `dummynet` трафик оформител, който се конфигурира с ipfw. Използват се потоци за натройка на лимитите на лентата в [K|M]{bit/s|Byte/s}, 0 означава неограничена лента. Използването на същия номер поток ще преконфигурира съществуващия. Например ограничаване на ъплоуд лентата до 500 Kbit.

```
# kldload dummynet                                 # зареждане на модула ако е необходимо
# ipfw pipe 1 config bw 500Kbit/s                  # създаване на поток с ограничена лента
# ipfw add pipe 1 ip from me to any                # отклоняване на пълния ъплоуд към потока
```

3.http://insecure.org/nmap/

QoS/Качество на услугата

Linux

Приоритезиране на опашката с `tc` за оптимизация на VoIP. Пълния пример може да се види на voip-info.org или www.howtoforge.com. Предполага се, че VoIP използва udp на портове 10000:11024 и устройство eth0 (може да бъде и ppp0 или so). Следващите команди дефинират QoS в три опашки и принуждават VoIP трафика в опашка 1 с QoS `0x1e` (всички битове са вдигнати). Трафика по подразбиране тече в опашка 3 и QoS *Minimize-Delay* тече в опашка 2.

```
# tc qdisc add dev eth0 root handle 1: prio priomap 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 0
# tc qdisc add dev eth0 parent 1:1 handle 10: sfq
# tc qdisc add dev eth0 parent 1:2 handle 20: sfq
# tc qdisc add dev eth0 parent 1:3 handle 30: sfq
# tc filter add dev eth0 protocol ip parent 1: prio 1 u32 \
  match ip dport 10000 0x3C00 flowid 1:1          # използва сървърния обхват портове
  match ip dst 123.23.0.1 flowid 1:1              # или/и използва сървърния IP
```

Статус и премахване с

```
# tc -s qdisc ls dev eth0                         # статус на опашката
# tc qdisc del dev eth0 root                      # изтриване на всички QoS
```

Калкулиране на обхвата портове и маска

tc филтъра дефинира обхват портове с порт и маска, които трябва да изчислите. Намерете 2^N *края* на набора портове, намалете с набора и конвертирайте в HEX. Това е маската. Пример за 10000 -> 11024, набора е 1024.

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

FreeBSD

Максималната лента на връзката е 500Kbit/s и дефинираме три опашки с приоритети 100:10:1 за VoIP:ssh:всичко останало.

```
# 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 използване на сървърен IP
# ipfw add 20 queue 2 dsp-port ssh
# ipfw add 30 queue 3 from me to any              # всичко останало
```

Статус и изтриване с

```
# ipfw list                                       # статус на правилата
# ipfw pipe list                                  # статус на потока
# ipfw flush                                      # изтриване на всички правила освен по подразби
```

4.13 NIS Дебъгване

Някой команди които биха работили на добре конфигуриран NIS клиент:

```
# ypwhich                 # показва името на NIS към който има връзка
# domainname              # NIS домейн името по конфигурация
# ypcat group             # показва групата от NIS сървъра
# cd /var/yp && make      # Построява наново yp базата данни
```

Работи ли ypbind?

```
# ps auxww | grep ypbind
/usr/sbin/ypbind -s -m -S servername1,servername2        # FreeBSD
/usr/sbin/ypbind          # Linux
# yppoll passwd.byname
```



```
# htpasswd -c /etc/svn-passwd user1  # -c създава файла
```

Примерен svn.acl за контрол на достъпа

```
# Default it read access. "* =" would be default no access
[/]
* = r
[groups]
project1-developers = joe, jack, jane
# Give write access to the developers
[project1:]
@project1-developers = rw
```

13.2 SVN команди и използване

Вижте също Subversion Quick Reference Card[10]. Tortoise SVN[11] е добър интерфейс за Windows.

Вмъкване

Нов проект, който е директория с файлове се вмъква в хранилището с командата `import`. Import също се използва и за добавяне на съдържание към съществуващ проект.

```
# svn help import                                 # Помощ за всяка команда
    # Добавя нова директория (със съдържание) в src директорията на project1
# svn import /project1/newdir http://host.url/svn/project1/trunk/src -m 'add newdir'
```

Стандартни SVN команди

```
# svn co http://host.url/svn/project1/trunk       # Изнасяне на последната версия
    # Етикети и разклонения са създадени при копирането
# svn mkdir http://host.url/svn/project1/tags/    # Създава tags директорията
# svn copy -m "Tag rc1 rel." http://host.url/svn/project1/trunk \
                                 http://host.url/svn/project1/tags/1.0rc1
# svn status [--verbose]                          # Проверява статуса на файловете в работната дир
# svn add src/file.h src/file.cpp                 # Добавя два файла
# svn commit -m 'Added new class file'            # Публикува промените със съобщение
# svn ls http://host.url/svn/project1/tags/       # Списък със всички етикети
# svn move foo.c bar.c                            # Премества (преименува) файлове
# svn delete some_old_file                        # Изтрива файлове
```

14 Полезни команди

14.1 less

Командата `less` показва текстов документ в конзолата. Налична е на повечето инсталации.

```
# less unixtoolbox.xhtml
```

Някои важни команди са (^N стои за [control]-[N]):

h H добра помощ на екрана
f ^F ^V SPACE Напред с един прозорец (за N реда).
b ^B ESC-v Назад с един прозорец (за N реда).
F Напред завинаги; като "tail -f".
/pattern Търси напред за (N-тия) съответстващ ред.

10.http://www.cs.put.poznan.pl/csobaniec/Papers/svn-refcard.pdf
11.http://tortoisesvn.tigris.org

13.1 Настройка на сървъра

Началото на сървъра е сравнително лесно (тук например трябва да съществура `/home/svn/`):

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

Достъпът до хранилището става възможен с:

- `file://` Директен достъп до файловата система чрез svn клиент. Това изисква локални права върху файловата система.
- `svn://` или `svn+ssh://` Отдалечен достъп чрез svnserve сървъра (също през SSH). Това изисква локални права върху файловата система.
- `http://` Отдалечен достъп с webdav, използвайки apache. За този и метод не са необходими локални потребители.

Използвайки локалната файлова система, дава възможност да бъдат внесен и после изнесен съществуващия проект. За разлика от CVS не е необходимо да влизате в директорията на проекта, просто задайте пълния път:

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

Новата директория "trunk" е само традиция и не е задължителна.

Отдалечен достъп със ssh

Не са необходими специални настройки за да имате достъп до хранилището през ssh, просто заменете `file://` със `svn+ssh/hostname`. На пример:

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

Както и при локален достъп, всеки потребител се нуждае от ssh достъп към сървъра (с локален акаунт). Този метод може да бъде прилаган при малки групи. Всички потребители могат да принадлежът към subverion група, на която принадлежи хранилището, на пример:

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

Отдалечен достъп през http (apache)

Отдалечен достъп през http (https) е добро решение само за голяма потребителска група. Този метод използва apache ауторизация, а не локални акаунти. Това е обикновена и малка конфигурация на 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    # Само за контрол на достъпа
```

```
<Location /svn>
  DAV svn
  # any "/svn/foo" URL will map to a repository /home/svn/foo
  SVNParentPath /home/svn
  AuthType Basic
  AuthName "Subversion repository"
  AuthzSVNAccessFile /etc/apache2/svn.acl
  AuthUserFile /etc/apache2/svn-passwd
  Require valid-user
</Location>
```

Apache сървърът се нуждае от пълен достъп до хранилището:

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

Създаване на потребител с htpasswd2:


```
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 broadcast
```

5 SSH SCP

5.1 Разпознаване чрез публичен ключ

Свързване с хост без парола с използване на публичен ключ. Идеята е да се дабави вашия публичен ключ към authorized_keys2 файла на отдалечения хост. За примера нека **свържем*хост-клиента* към *хост-сървъра***, ключа е генериран на клиента.

- Използвайте ssh-keygen за генериране на двойка ключове. `~/.ssh/id_dsa` е частния ключ, `~/.ssh/id_dsa.pub` е публичния.
- Копирайте само публичния ключ върху сървъра и го добавете към файла `~/.ssh/authorized_keys2` във вашата домашна директория на сървъра.

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

Използване на Windows клиент от ssh.com

Некомерсиална версия на ssh.com клиент може да бъде свлена от главния ftp сайт: ftp.ssh.com/pub/ssh/. Ключовете генерирани от ssh.com клиента трябва да бъдат конвертирани за OpenSSH сървър. Това може да бъде направено с ssh-keygen команда.

- Създаване на двойка ключове с ssh.com клиента: Settings - User Authentication - Generate New....
- Използвам ключ тип DSA; дължина 2048.
- Копираме публичния ключ генериран от ssh.com клиента на сървъра в ~/.ssh папката.
- Ключовете за в C:\Documents and Settings\%USERNAME%\Application Data\SSH\UserKeys.
- Използваме ssh-keygen команда на сървъра за конвертиране на ключа:

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

Забележка: Използваме DSA ключ, RSA също е възможен. Ключа няма парола за защита.

Използване на putty за Windows

Putty[4] е прост и с отворен код ssh клиент за Windows.

- Създаване на двойка ключове с puTTYgen програмата.
- Записваме публичния и частния ключ (например в C:\Documents and Settings\%USERNAME%\.ssh).
- Копираме публичния ключ на сървъра в ~/.ssh директорията:

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

- Използваме ssh-keygen командата на сървъра за конвертиране на ключа за OpenSSH:

4.http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

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

- Посочваме мястото на частния ключ в настройките на putty : Connection - SSH - Auth

5.2 Проверка на оптечатък

При първи логин, ssh ще пита дали непознат хост чрез неговия отпечатък да бъде записан към познатите хостове. За избягване човек-по-средата (man-in-the-middle) атака администратора на сървъра мое да ви изпрати отпечатъка на сървъра, който да сравните при първи логин. Използвайте `ssh-keygen -l` да получите отпечатък (на сървъра):

```
# ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub      # За RSA ключ
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      # За DSA ключ (подразбиране)
2048 14:4a:aa:d9:73:25:46:6d:0a:48:35:c7:f4:16:d4:ee /etc/ssh/ssh_host_dsa_key.pub
```

Сега клиента свързващ се към този сървър може да провери че се свързва към правилния сървър:

```
# ssh linda
The authenticity of host 'linda (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
```

5.3 Сигурен трансфер на файлове

Някой прости команди:

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

В Konqueror или Midnight Commander е възможно да се достигне отдалечена файлова система с адрес **fish://user@gate**. Обаче скоростта е много ниска.
Освен това е възможно да се монтира отдалечена дирекория със **sshfs**, клиент за файлова система, базиран на SCP. Вижте fuse sshfs[5].

5.4 Тунелиране

SSH тунелирането позволява да изпратим или реверсивно изпратим порт през SSH връзка, по този начин защитавайки трафика и достъпвайки портове, блокирани иначе. Това работи само под TCP. Основния начин за изпращане и реверс е (виж същоssh и NAT пример):

```
# ssh -L localport:desthost:destport user@gate  # хост-дестинация видян от гейта
# ssh -R destport:desthost:localport user@gate  # изпраща вашия локален порт към дестинация
# ssh -X user@gate   # Форсиране на X изпращане
```

Това ще се свърже към гейта и ще изпрати локален порт към хоста desthost:destport. Забележете desthost е хоста дестинация *видян откъм гейта*, така че ако връзката е към гейта, то desthost е localhost. Може да се изпрати повече от един порт.

Директно изпращане към гейта

Да кажем искаме достъп до CVS (port 2401) и http (port 80) работещи на гейта. Това е най-прост пример, desthost е следователно localhost, и използваме порт 8080 локално вместо 80 така че няма нужда да сме root. След като ssh сесията е отворена, и двете услуги са достъпни на локалните портове.

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

5.http://fuse.sourceforge.net/sshfs.html


```
CVS password:
# cvs checkout MyProject/src
```

12.4 CVS команди и използване

Внасяне

Командата import се използва за добавяне на цяла директория, трябва да бъде стартирана от директорията която ще бъде внесена. Да речем директорията /devel/ съдържа всички файлове и поддиректории, които да бъдат внесени. Името на директорията в CVS (модулът) ще бъде наречена "myapp".

```
# cvs import [options] directory-name vendor-tag release-tag
# cd /devel                          # Трябва да бъде в проекта, за да бъде внесен
# cvs import myapp Company R1_0      # Release tag може да бъде всичко в една дума
```

След време беше добавена нова директория "/devel/tools/" и трябва също да бъде внесена.

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

Изнасяне обновяване добавяне изпълнение

```
# cvs co myapp/tools                 # Ще изнесе само директорията tools
# cvs co -r R1_1 myapp               # Изнася myapp във версия R1_1 (залепено е)
# cvs -q -d update -P                # Типично CVS обновяване
# cvs update -A                      # Поправя всеки залепнал етикет (или дата, опция)
# cvs add newfile                    # Добавя нов файл
# cvs add -kb newfile                # Добавя нов изпълним файл
# cvs commit file1 file2             # Предава само последните два файла
# cvs commit -m "съобщение"            # Предава всички направени промени със съобщение
```

Създаване на кръпка

Най-добре е да създадете и приложите кръпка от работната разработваща директория свързана с проекта, или от самата директория съдържаща източника.

```
# cd /devel/project
# diff -Naur olddir newdir > patchfile # Създава кръпка от директория или файл
# diff -Naur oldfile newfile > patchfile
```

Прилагане на кръпка

Понякога е необходимо да свалите нивото на директория от кръпката, в зависимост от това как е бил създаден. В случай на трудности, просто погледнете първите редове на кръпката и опитайте -p0, -p1 или -p2.

```
# cd /devel/project
# patch --dry-run -p0 < patchfile    # Тествайте кръпката без да бъде прилагана
# patch -p0 < patchfile
# patch -p1 < patchfile              # Смъква 1-вото ниво от кръпката
```

13 SVN

Настройка на сървър (p36) | SVN+SSH (p36) | SVN през http (p36) | SVN използване (p37)

Subversion (SVN)[8] е система за контролиране на версията, създадена да бъде заместникът на CVS (Система за конкурентни версии). Концептът е подобен на CVS, но много недостатъци са изчистени. Вижте също SVN книгата[9].

8.http://subversion.tigris.org/
9.http://svnbook.red-bean.com/en/1.4/

Сега добавете `:cvs` в края на всеки ред, което казва на cvs сървъра да промени потребителя в cvs (или под каквото и да работи вашият cvs сървър). Изглежда така:

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

12.2 Пробвайте

Пробвайте входа като обикновен потребител (като пример тук - аз)

```
# 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:
```

CVSROOT променлива

Това е променлива на средата използвана за определяне на позицията на хранилището, в което изпълняваме операции. За локална употреба, може да бъде просто зададено като директория в хранилището. За използване в мрежата, транспортния протокол трябва да бъде определен. Задайте CVSROOT променливата с `setenv CVSROOT string` в csh, tcsh shell, или `export CVSROOT=string` в sh, bash shell.

```
# setenv CVSROOT :pserver:<username>@<host>:/cvsdirectory
На пример:
# setenv CVSROOT /usr/local/cvs                                  # Използва се само локално
# setenv CVSROOT :local:/usr/local/cvs                           # Същото като горното
# setenv CVSROOT :ext:user@cvsserver:/usr/local/cvs              # Директен достъп през SSH
# setenv CVS_RSH ssh                                             # За ext достъпа
# setenv CVSROOT :pserver:user@cvsserver.254:/usr/local/cvs      # Мрежа с pserver
```

Когато входът е успешен един може да вмъкне нов проект в хранилището: **cd into** основната директория на вашия проект

```
cvs import <module name> <vendor tag> <initial tag>
cvs -d :pserver:colin@192.168.50.254:/usr/local/cvs import MyProject MyCompany START
```

Където MyProject е името на новия проект в хранилището (използвано по-късно за изнасяне). Cvs ще вмъкне съдържанието на настоящата директория в новият проект.

За изнасяне:

```
# 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
```

12.3 SSH тунелиране за CVS

За това ни трябват два шела. В първият шел се свързваме към cvs сървъра със ssh и порт-препращаме cvs връзката. На втория шел използваме cvs нормално както когато се работи локално.
на шел 1:

```
# ssh -L2401:localhost:2401 colin@cvs_server    # Свързване директно към CVS сървъра. Или:
# ssh -L2401:cvs_server:2401 colin@gateway      # Използва врата (gateway) за достигане на CVS
```

на шел 2:

```
# setenv CVSROOT :pserver:colin@localhost:/usr/local/cvs
# cvs login
Logging in to :pserver:colin@localhost:2401/usr/local/cvs
```


Netbios и отдалечен десктоп изпращане към втори сървър

Да кажем Windows smb сървър е зад гейт и няма ssh. Трабва да имаме достъп до smb споделяне и отдалечен десктоп до сървъра.

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

smb споделянето сега може да бъде достъпено с \\127.0.0.1\, но само ако локалното споделяне е изключено, защото*локалното споделяне слуша на порт 139*.
Възможно е да имаме също и локално сподеялне, за това трябва да създадем ново виртуално устройство с нов IP адрес за тунела, smb споделянето ще бъде свързано през този адрес. Нататък *локалния RDP (отдалечен десктоп) вече слуша на порт 3389*, така че избираме 3388. За примера да използваме виртуален IP 10.1.1.1.

- Със putty използваме Source port=10.1.1.1:139. Възможно е да се създадат мнжожество циклови устройства и тунел. На Windows 2000, само putty работеше.
- Със ssh.com клиента, изключете "Allow local connections only". След като ssh.com ще се свързва към всички адреси, само единично споделяне може да се свърже.

Сега създаваме обратен интерфейс със 10.1.1.1:

- # System->Control Panel->Add Hardware # Yes, Hardware is already connected # Add a new hardware device (към края).
- # Install the hardware that I manually select # Network adapters # Microsoft , Microsoft Loopback Adapter.
- Конфигурираме IP адрес на фалшиво устройство 10.1.1.1 маска 255.255.255.0, без гейт.
- advanced->WINS, Enable LMHosts Lookup; Disable NetBIOS over TCP/IP.
- # Enable Client for Microsoft Networks. # Disable File and Printer Sharing for Microsoft Networks.

ТРЯБВАШЕ да рестартирам за да заработи. Сега се свързваме към smb споделянето с \\10.1.1.1 и отдалечен десктоп към 10.1.1.1:3388.

Debug

Ако не работи:

- Изпратени ли са портовете: netstat -an? Вижте 0.0.0.0:139 или 10.1.1.1:139
- Свързва ли се telnet 10.1.1.1 139 ?
- Трябва да поставите отметка "Local ports accept connections from other hosts".
- Is "File and Printer Sharing for Microsoft Networks" изключено ли е на обратния интерфейс?

Свързване на два клиента зад NAT

Да допуснем два клиента са зад NAT гейт и клиента cliadmin трябва да се свърже с клиента cliuser (дестинация), и двата могат да влизат в гейта с ssh и работят с Linux със sshd. Нямате нужда от root достъп никъде, докато портовете на гейта са над номер 1024. Използваме 2022 на гейта. След като гейта е използван локално, опцията GatewayPorts не е необходима.
На клиента cliuser (от посока към гейта):

```
# ssh -R 2022:localhost:22 user@gate            # изпраща клиент 22 към гейта:2022
```

На клиента cliadmin (от хоста към гейта):

```
# ssh -L 3022:localhost:2022 admin@gate         # изпраща клиент 3022 към гейта:2022
```

Сега администратора може да се свърже директно към клиента cliuser със:

```
# ssh -p 3022 admin@localhost                   # локален:3022 -> гейт:2022 -> клиент:22
```

Свързване към VNC зад NAT

Да предположим Windows клиент със VNC слушащ на порт 5900 трябва да бъде достъпен зад NAT. На клиента cliwin към гейта:

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

На клиента cliadmin (от хоста към гейта):

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

Сега администратора може да се свърже директно към клиентския VNC със:

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

6 VPN със SSH

От версия 4.3, OpenSSH може да ползва tun/tap устройство за криптиране на тунел. Това е много подобно на други TLS базирани VPN решения като OpenVPN. Едно предимство на SSH е, че не е необходимо да се инсталира и конфигурира допълнителен софтуер. Допълнително тунелът използва SSH разпознаване, например споделени ключове. Недостатък е капсулирането през TCP, което може да доведе до проблеми при бавна връзка. Също така тунелът разчита на единична ("крехка") TCP връзка. Тази техника е много практична за бързо IP базирано VPN решение. Няма ограничение като при единично TCP порт препращане, всички протоколи слой 3/4 като ICMP, TCP/UDP, и т.н. са препратени през VPN. Във всеки случай са нужни следните опции в sshd_conf файла:

```
PermitRootLogin yes
PermitTunnel yes
```

6.1 Единична P2P връзка

Тук свързваме два хоста, hclient и hserver със peer to peer тунел. Връзката *стартира от hclient* към hserver и е под потребител root. Крайните точки на тунела са 10.0.1.1 (сървър) и 10.0.1.2 (клиент) и създаваме устройство tun5 (или друг номер). Процедурата е проста:

- Връзка с SSH с използване на тунелната опция -w
- Конфигуриране на IP адреса на тунела. Веднаж на сървъра и веднаж на клиента.

Връзка към сървъра

Връзката се стартира на клиента и командите се изпълняват на сървъра.

Сървърът е на Linux

```
cli># ssh -w5:5 root@hserver
srv># ifconfig tun5 10.0.1.1 netmask 255.255.255.252   # Изпълнено в сървърен шел
```

Сървърът е на FreeBSD

```
cli># ssh -w5:5 root@hserver
srv># ifconfig tun5 10.0.1.1 10.0.1.2                  # Изпълнено в сървърен шел
```

Конфигуриране на клиента

Команди изпълнени на клиента:

```
cli># ifconfig tun5 10.0.1.2 netmask 255.255.255.252   # Клиента е на Linux
cli># ifconfig tun5 10.0.1.2 10.0.1.1                  # Клиента е на FreeBSD
```

Двата хоста сега са свързани и могат прозрачно да комуникират със всеки протокол от 3/4 слой, използвайки тунелни IP адреси.

6.2 Свързване на две мрежи

В допълнение на p2p настройката по-горе, е много полезно свързването на две частни мрежи с SSH VPN с използване на два гейта. Да допуснем имаме мрежаA 192.168.51.0/24 и мрежаB 192.168.16.0/24. Процедурата е подобна на горната, единствено добавяме рутиране. NAT трябва да е активирано на частен интерфейс само ако гейтовете не са


```
# cvs init                              # Създава всички вътрешни CVS конфигурационни файлове
# cd /root
# cvs checkout CVSROOT                  # Проверява конфигурационните файлове, за да ги промени
# cd CVSROOT
edit config (добре е катко си е)
# cvs commit config
cat >> writers                          # Създава файл на писателите (допълнително може и на читател
colin
^D                                      # Използвайте [Control][D] за да излезете от редактирането
# cvs add writers                       # Добавя файла writers към хранилището
# cvs edit checkoutlist
# cat >> checkoutlist
writers
^D                                      # зползвайте [Control][D] за да излезете от редактирането
# cvs commit                            # Извършва всички конфигурационни промени
```

Добавете **readers** файл ако искате да дефинирате права за четене и писане *Забележка:* (Никога) Не променяйте файлове директно в главния cvs, но по-скоро изнесете файла, променете го и го внесете. Направихме това с файла **writers** за да дефинираме достъпа за писане.
До този момент има три популярни начина за достъп до CVS. Първите два не се нуждаят от допълнителни настройки. Вижте примерите в CVSROOT по-долу, за това как да ги използвате:

- Директен локален достъп до файловата система. Потребителя(те) се нуждае/ят от подходящи права върху файловете, за да имат достъп до CVS директно и няма по-нататъчна ауторизация в допълнение към влизането в операционната система. Все пак това е полезно ако хранилището е локално.
- Отдалечен достъп през ssh с ext протокола. Всеки потребител със ssh shell акаунт и права за четене/писане на CVS сервъра може да има директен достъп до CVS с ext през ssh без допълнителен тунел. Не е необходим сървърен процес работещ на CVS за да работи това. Входът през ssh изпълнява ауторизацията.
- Отдалечен достъп през pserver. Това не с препоръчва за използване, при голяма база от потребители, тъй като потребителите са ауторизиране от CVS pserver със съответна база данни за паролите, затова и няма нужда от локални потребителски акаунти. Тези настройки са обяснени отдолу.

Мрежова настройка с inetd

CVS може да бъде стартиран локално само ако достъпа до мрежата не е необходим. За отдалечен достъп, демона inetd може да стартира pserver със следния ред в /etc/inetd.conf (/etc/xinetd.d/cvs на SuSE):

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

Добра идея е да блокирате cvs порта от Интернет със огнената стена и да използвате ssh тунел за отдалечен достъп до хранилището.

Отделна ауторизация

Възможно е да имате cvs потребители, които не са част от операционната система (не са локални потребители). Това всъщност вероятно е и желано от гледна точка на сигурността. Просто добавете файл озаглавен **passwd** (в CVSROOT директорията) съдържащ потребителското име и паролата в криптиран формат. Това може да бъде направено с htpasswd инструмента на Apache.
Забележка: Този passwd файл е единственият файл, който трябва да бъде променен директно в CVSROOT директорията. Също няма да бъде изнасян. Повече информация с htpasswd --help

```
# htpasswd -cb passwd user1 password1  # -c създава файла
# htpasswd -b passwd user2 password2
```

11.5 Подписване на сертификата

Изискването за сертификат трябва да бъде подписано от CA, за да е валидно, Тази стъпка обикновено се изпълнява от продавача на сертификата. *Забележка: заменете "servername" с името на вашия сървър в следващите команди*.

```
# 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
```

Сега servernamekey.pem е личния ключ и servernamecert.pem сертификата на сървъра.

11.6 Създаване на обединен сертификат

IMAP сървъра иска личния ключ и сертигиката на сървъра да бъдат в един файл. А главно това е и по-лесно за работа, но файла трябва да бъде защитен!. Apache също може добре да работи с него. Създайте файл servername.pem съдържащ и двете - сертификата и ключа.

- Отворете личния ключ (servernamekey.pem) с текстов редактор и копирайте личния ключ в "servername.pem" файла.
- Направете същото със сертификата на сървъра (servernamecert.pem).

Крайния servername.pem file би трябвало да изглежда така:

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

Какво имаме сега в директорията /usr/local/certs/:

CA/private/cakey.pem *(CA личен ключ на сървъра)*
CA/cacert.pem *(CA личен ключ на сървъра)*
certs/servernamekey.pem *(личен ключ на сървъра)*
certs/servernamecert.pem *(подписан сертификат на сървъра)*
certs/servername.pem *(сертификат с ключ на сървъра)*

Пазете личния клюя защитен!

11.7 Преглед на информацията за сертификата

За да видите информацията за сертификата просто изпълнете:

```
# openssl x509 -text -in servernamecert.pem      # Преглед на информацията за сертификата
# openssl req -noout -text -in server.csr        # Преглед на информацията за запитването
```

12 CVS

Настройка на сървър (p32) | CVS тест (p34) | SSH тунелиране (p34) | CVS употреба (p35)

12.1 Настройка на сървър

Лансиране на CVS

Решете къде главният източник ще складира и създаде основен cvs. Например /usr/local/cvs (като основа):

```
# mkdir -p /usr/local/cvs
# setenv CVSROOT /usr/local/cvs      # Задава CVSROOT към новото положение (локално)
```


гейтовете по подразбиране за техните мрежи.
192.168.51.0/24 (мрежаA)|гейтA <-> гейтB|192.168.16.0/24 (мрежаB)

- Връзка със SSH с използване на тунелната опция -w.
- Конфигурираме IP адреса на тунела. Веднаж на сървъра и веднаж на клиента.
- Добавяме рутиране за двете мрежи.
- Ако е необходимо, активираме NAT на частния интерфейс на гейта.

Конфигурацията се *стартира от гейтA във мрежаA*.

Връзка от гейтA до гейтB

Връзката е започната от гейтA, командите се изпълняват на гейтB.

гейтB е на Linux

```
gateA>#  ssh -w5:5 root@gateB
gateB>#  ifconfig tun5 10.0.1.1 netmask 255.255.255.252 # Изпълнено в шел-а на гейтB
gateB>#  route add -net 192.168.51.0 netmask 255.255.255.0 dev tun5
gateB>#  echo 1 > /proc/sys/net/ipv4/ip_forward         # Само ако гейта не е дефолт
gateB>#  iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

гейтB е на FreeBSD

```
gateA>#  ssh -w5:5 root@gateB                           # Създава tun5 устройство
gateB>#  ifconfig tun5 10.0.1.1 10.0.1.2                # Изпълнено в шел-а на гейтB
gateB>#  route add 192.168.51.0/24 10.0.1.2
gateB>#  sysctl net.inet.ip.forwarding=1                # Само ако гейта не е дефолт
gateB>#  natd -s -m -u -dynamic -n fxp0                 # виж NAT (page 16)
gateA>#  sysctl net.inet.ip.fw.enable=1
```

Конфигуриране на гейтA

Команди, изпълнени на гейтA:

гейтA е 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
```

гейтA е 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                 # виж NAT (page 16)
gateA>#  sysctl net.inet.ip.fw.enable=1
```

Двете частни мрежи сега са прозрачно свързани през SSH VPN. IP препращането и NAT настройките са необходими само ако гейтовете не са по подразбиране (дефолт). В този случай клиентите не биха знали къде да препратят заявка, заради което трябва да се активира NAT.

7 RSYNC

Rsync може почти напълно да замени cp и scp, допълнително прекъснатите трансфери ще бъдат ефективно рестартирани. Последната наклонена (/) (и нейното отсъствие) имат различно действие, man страницата е добра... Някой примери:
Копиране на директории със съдържание:

```
# rsync -a /home/colin/ /backup/colin/
# rsync -a /var/ /var_bak/
# rsync -aR --delete-during /home/user/ /backup/      # използване на относителност (виж надолу)
```

Като предишното, но по мрежа и с компресия. Rsync използва SSH за транспорт по подразбиране и ще ползва ssh ключ ако има. Използвайте ":" като с SCP. Типично отдалечено копиране:

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

Изключване на всяка директория tmp във /home/user/ и съхраняване на иерархията на директориите, т.е. отдалечената директория ще има структура /backup/home/user/. Типично се използва за бекъп.

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

Използване на порт 20022 за ssh връзка:

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

Използване на rsync daemon (със "::") е много по-бързо, но не е криптирано с ssh. Мястото на /backup е дефинирано от конфигурацията в /etc/rsyncd.conf. Променливата RSYNC_PASSWORD може да бъде дефинирана за избягване ръчното и въвеждане.

```
# rsync -axSRz /home/ ruser@hostname::rmodule/backup/
# rsync -axSRz ruser@hostname::rmodule/backup/ /home/    # Копиране назад
```

Важни опции:

`-a, --archive` архивен режим; като -rlptgoD (без -H)
`-r, --recursive` рекурсивно в директориите
`-R, --relative` използване на относителни имена
`-H, --hard-links` запазване на хард-връзките
`-S, --sparse` ефективна работа със пръснати файлове
`-x, --one-file-system` без да напуска файловата система
`--exclude=PATTERN` изключва файлове съвпадащи с PATTERN
`--delete-during` получателя изтрива по време на трансфера, не преди
`--delete-after` получателя изтрива след трансфера, не преди

7.1 Rsync на Windows

Rsync е достъпен за Windows през cygwin или като самостоятелен пакет от cwrsync[6]. Много е удобно за автоматичен бекъп. Инсталирайте един от двата (*не и двата*) и добавете пътя до системните променливи на Windows: # Control Panel -> System -> табът Advanced, бутон Environment Variables. Редактирайте "Path" променливата и добавете пълния път до инсталирания rsync, т.е. C:\Program Files\cwRsync\bin или C:\cygwin\bin. По този начинкомандите `rsync` и `ssh` ще са достъпни в Windows команден шел.

Идентификация с публичен ключ

Rsync автоматично се тунелира в SSH и следователно използва SSH идентификация на сървъра. Автоматичните бекъпи трябва да избягнат намесата на потребителя, за това SSH идентификация по публичен ключ може да се използва и rsync командата ще работи без парола.
Всички следващи команди се изпълняват във Windows конзола. В конзолата (Start -> Run -> cmd) създайте и изпратете ключа както е описано в SSH, променете "user" и "server" със необходимите стойности. Ако файла authorized_keys2 още не съществува, просто копирайте id_dsa.pub в authorized_keys2 и го изпратете.

```
# ssh-keygen -t dsa -N ''                    # Създава пбличен и частен ключ
# rsync user@server:.ssh/authorized_keys2 .  # Копирайте файла локално от сървъра
# cat id_dsa.pub >> authorized_keys2         # Или използвайте редактор за добавяне на ключа
# rsync authorized_keys2 user@server:.ssh/   # Копиране файловете обратно на сървъра
# del authorized_keys2                       # Изтриване на локалното копие
```

Сега тест (на един ред):

6.http://sourceforge.net/projects/sereds

11.1 Процедура

- Трябва ни сертификаторско пълномощно, за да подпишем нашия сертификат. Тази стъпка обикновенно се предлага от продавачи като Thawte, Verisign и т.н., но ние все пак можем да си направиме собствено.
- Създаване на изискване за подпис на сертификат. Това изискване е като неподписан сертификат (публичната част) и вече съдържа всичката нужна информация. Изискването за сертификат обикновено се праща до продавача на пълномощни за подпис. Тази стъпка също така създава и личния ключ на локалната машина.
- Подписване на сертификата с пълномощно за сертификата.
- Ако е необходимо съединете сертификата и ключа в един файл, за да бъде използван от приложение (уеб-сървър, мейл сървър и т.н.).

11.2 Конфигуриране на OpenSSL

Използваме /usr/local/certs като директория за тази примерна проверка или променяме /etc/ssl/openssl.cnf съответващо на вашите настройки, за да знаете къде ще бъдат създадени файловете. Това е важната част от openssl.cnf:

```
[ CA_default ]
dir             = /usr/local/certs/CA       # Къде се съхранява всичко
certs           = $dir/certs                # Къде се съхранява issued certs
crl_dir         = $dir/crl                  # Къде се съхранява issued crl
database        = $dir/index.txt            # Индекс файл на базата данни.
```

Уверете се че директориите съществуват и ако трябва ги създайте

```
# mkdir -p /usr/local/certs/CA
# cd /usr/local/certs/CA
# mkdir certs crl newcerts private
# echo "01" > serial                        # Само ако serial не същестува
# touch index.txt
```

11.3 Създаване на сертификат пълномощно

Ако нямате пълномощно за сертификат от продаващ такива, трябва да си зъздадете собствено. Тази стъпка не е необходима, за онези решили да използват продавач, който да подпише изискването. За да направите пълномощно за сертификат (CA):

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

11.4 Създаване на изискване за подпис на сертификата

За да направите нов сертификат (за мейл сървър или уеб сървър например), първо създайте изискване за сертификат с неговия личен ключ. Ако вашето приложение не поддържа криприран личен ключ (на пример UW-IMAP не поддържа), тогава забранете криптирането с `-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 криптиране на ключа
```

`/root/ad1.key`, за да прикачите дялът. Главният ключ е записан в дяла и не е достъпен. Виж по-долу за типично USB или файлово базирана снимка.

Създаване на криптиран дял

```
# dd if=/dev/random of=/root/ad1.key bs=64 count=1  # този ключ криптира главният ключ
# geli init -s 4096 -K /root/ad1.key /dev/ad1       # -s 8192 също е добре за дискове
# geli attach -k /root/ad1.key /dev/ad1             # ДА направете бекъп на /root/ad1.key
# dd if=/dev/random of=/dev/ad1.eli bs=1m           # Незадължително и отнема повече време
# newfs /dev/ad1.eli                                # Създава файлова система
# mount /dev/ad1.eli /mnt
```

Прикачване

```
# geli attach -k /root/ad1.key /dev/ad1
# fsck -ny -t ffs /dev/ad1.eli                      # При съмнение проверете файловата система
# mount /dev/ad1.eli /mnt
```

Откачане

Процедурата за откачване става автоматично при изключване.

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

/etc/fstab

Криптираният дял може да бъде конфигуриран, за да бъде прикачен с /etc/fstab. Паролата ще бъде поискане при зареждане. За този пример са необходими следните настройки:

```
# 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
```

Използване само на парола

По подходящо е да криптиране USB памет или файлово базирана снимка с парола и без ключ. В този случай не е нобходимо да носите допълнителен файл с ключ наоколо. Процедурата е почти същата като по-горе, просто без ключовия файл. Нека криптираме файлово базирана снимка `/cryptedfile` от 1 GB.

```
# dd if=/dev/zero of=/cryptedfile bs=1M count=1000  # 1 GB file
# mdconfig -at vnode -f /cryptedfile
# geli init /dev/md0                                # криптира само с парола
# geli attach /dev/md0
# newfs -U -m 0 /dev/md0.eli
# mount /dev/md0.eli /mnt
# umount /dev/md0.eli
# geli detach md0.eli
```

Сега е възможно да прикачите тази снимка на друга файлова система само с паролата.

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

11 SSL Сертификати

Тъй наречените SSL/TLS сертификати са криптографични публични ключове сертификати и се състоят от публичен и личен ключ. Сертификатите се използва за удостоверяване на крайни точки и криптиране на данни. Използват се например на уеб-сървър (https) или мейл сървър mail server (imaps).


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

Автоматичен бекъп

Използвайте batch файл за автоматизиране на бекъпа и го добавете към разсточените задачи (Programs -> Accessories -> System Tools -> Scheduled Tasks). За пример създайте файла backup.bat и заменете user@server.

```
@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
```

8 SUDO

Sudo е стандартен начин, да дадете на потребителите някои администраторски права, без да издава администраторската парола. Sudo е много полезен в многопотребителска среда с микс от сървър и работни станции. Просто извикайте команда със sudo:

```
# sudo /etc/init.d/dhcpd restart            # Изпълнява rc скрипта като root
# sudo -u sysadmin whoami                   # Изпълнява cmd като друг потребител user
```

8.1 Конфигурация

Sudo е конфигуриран в `/etc/sudoers` може да бъде променен само с `visudo`. Основния синтаксис е (списъците са разделени със запетая):

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

`users` един или повече потребители, или %group (като %wheel) да получат правата
`hosts` списък на хостовете (или ALL)
`runas` списък на потребителите (или ALL) като които правилото за команда може да бъде изпълнено. Оградено е със ( )!
`commands` списък на командите (или ALL) които ще бъдат изпълнени като root или като (runas)

Допълнително тези ключови думи могат да бъдат дефинирани като название (alias), те са наречени User_Alias, Host_Alias, Runas_Alias и Cmnd_Alias. Това е полезно при по-големи конфигурации. Ето един sudoers пример:

```
# cat /etc/sudoers
# Хост наимернования са subnets или hostnames.
Host_Alias   DMZ     = 212.118.81.40/28
Host_Alias   DESKTOP = work1, work2

# Потребителски наименования са списък от потребители, които могат да имат еднакви права
User_Alias   ADMINS  = colin, luca, admin
User_Alias   DEVEL   = joe, jack, julia
Runas_Alias  DBA     = oracle,pgsql

# Команда наименование дефинира пълния път до списък с команди
Cmnd_Alias   SYSTEM  = /sbin/reboot,/usr/bin/kill,/sbin/halt,/sbin/shutdown,/etc/init.d/
```

```
Cmnd_Alias    PW      = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root # Not root pwd!
Cmnd_Alias    DEBUG   = /usr/sbin/tcpdump,/usr/bin/wireshark,/usr/bin/nmap
```

```
# Актуалните права
root,ADMINS   ALL     = (ALL) NOPASSWD: ALL     # ADMINS могат да правят всичко с/без парола.
DEVEL         DESKTOP = (ALL) NOPASSWD: ALL     # Разработчици има пълни права на десктопи
DEVEL         DMZ     = (ALL) NOPASSWD: DEBUG   # Разработчици могат да дебъгват DMZ сървърите.

# Потребител sysadmin (системен администратор) може да се бърка в DMZ сървърите с някои команди.
sysadmin      DMZ     = (ALL) NOPASSWD: SYSTEM,PW,DEBUG
sysadmin      ALL,!DMZ = (ALL) NOPASSWD: ALL    # Може да прави всичко извън DMZ.
%dba          ALL     = (DBA) ALL               # Група dba може да стартира като потребител на баз

# Всеки може да прикачва/откача cd-устройство на десктоп машини
ALL           DESKTOP = NOPASSWD: /sbin/mount /cdrom,/sbin/umount /cdrom
```

9 Криптиране на файлове

9.1 Отделен файл

Криптиране и декриптиране:

```
# openssl des -salt -in file -out file.des
# openssl des -d -salt -in file.des -out file
```

Забележете, че файла разбира се може да бъде и tar архив.

9.2 tar и критпиране на цяла директория

```
# tar -cf - directory | openssl des -salt -out directory.tar.des      # Криптира
# openssl des -d -salt -in directory.tar.des | tar -x                 # Декриптира
```

9.3 tar zip и криптиране на цяла директория

```
# tar -zcf - directory | openssl des -salt -out directory.tar.gz.des  # Криптира
# openssl des -d -salt -in directory.tar.gz.des | tar -xz             # Декриптира
```

- Използвайте -k моятатайнапарола след des, за да избегнете интерактивното запитване за парола. Въпреки това имайте предвид, че това е много несигурно.
- Използвайте **des3** вместо **des** за да получите дори по-силно криптиране (Triple-DES Cipher). Това използва и повече CPU.

10 Криптиране на дялове

Linux с LUKS (p29) | Linux само dm-crypt (p29) | FreeBSD GELI (p29) | FBSD само pwd (p30)

Има (много) други алтернативни методи за криптиране на дискове, аз показвам само методите, които знам и използвам. Имайте предвид, че сигурността е само тогава добра, когато операционната система не е калена с това. Нарушител може лесно да запише паролата от клавиатурните събития. Още повече данните са свободно достъпни когато дялът е *прикрепена* и на този етап няма да откаже достъп до нея на нарушител.

10.1 Linux

Тези инструкции използват `dm-crypt` в Linux (device-mapper) удобства налични в 2.6 ядро. В този пример нека криптираме дяла `/dev/sdc1`, това може да бъде и всеки друг дял от диска или USB, или файлово базиран дял, създаден с `losetup`. В този случай ще

използваме `/dev/loop0`. Виж файл снимка на дял. Картографът на устройствата използва етикети за идентифициране на дял. Ние използваме `sdc1` в този пример, но това може да бъде всеки стринг.

dm-crypt с LUKS

LUKS с dm-crypt има по-добро криптиране и прави възможно да имаме няколко пароли за един и същи дял или лесно да промениме паролата. За да проверил дали LUKS е достъпен, просто напишете `# cryptsetup --help`, ако не се появи нищо относно LUKS, използвайте следните инструкции Без LUKS. Първо създайте дял ако е необходим: `fdisk /dev/sdc`.

Създаване на криптиран дял

```
# dd if=/dev/urandom of=/dev/sdc1           # Незадължително. Само за параноиди (отнема дни)
# cryptsetup -y luksFormat /dev/sdc1        # Това унищожава всякакви данни на sdc1
# cryptsetup luksOpen /dev/sdc1 sdc1
# mkfs.ext3 /dev/mapper/sdc1                # създава ext3 файлова система
# mount -t ext3 /dev/mapper/sdc1 /mnt
# umount /mnt
# cryptsetup luksClose sdc1                 # Откача криптирания дял
```

Прикачване

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

Откачане

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

dm-crypt без LUKS

```
# cryptsetup -y create sdc1 /dev/sdc1       # или всеки друг дял като /dev/loop0
# dmsetup ls                                # проверете, ще покаже: sdc1 (254, 0)
# mkfs.ext3 /dev/mapper/sdc1                # Това се прави само първият път!
# mount -t ext3 /dev/mapper/sdc1 /mnt
# umount /mnt/
# cryptsetup remove sdc1                    # Откача криптираният дял
```

Направете абсолютно същото (без mkfs частта!) за да пре-прикачите дяла. Ако паролата е грешна, mount командата няма да бъде изпълнена. В този случай просто премахнете sdc1 (`cryptsetup remove sdc1`) и го създайте пак.

10.2 FreeBSD

Двата популярни FreeBSD модула за криптиране на дискове са `gbde` и `geli`. Аз използвам geli, защото е по-бърз и освен това използва crypto устройства за хардуерно ускорение. Виже FreeBSD handbook Chapter 18.6[7] за всички детайли. geli модулът трябва да бъде зареден или компилиран в ядрото:

```
options GEOM_ELI
device crypto                                       # или като module:
# echo 'geom_eli_load="YES"' >> /boot/loader.conf   # или изпълнете: kldload geom_eli
```

Използване на парола и ключ

Аз използвам тези настройки за типично криптиране на диск, използва парола И ключ за криптиране на главния ключ. Това е, тябват ви двете - паролата и генерирания ключ

7.http://www.freebsd.org/handbook/disks-encrypting.html