user And LINUX

Больше чем user

№ 3 МАРТ 2011

Настройка PPPoE с шейпингом трафика для небольшой сети

Интсрукция по серверу Ejabberd

Access Control List: списки контроля доступа

urpmi: Краткое руководство пользователя

Laptop Mode Tools: уменьшаем энергопотребление ноутбука

Управление несколькими компьютерами одной клавиатурой и мышью

Vim: необходимый минимум знаний

UA Linux
open source ■ open future

Тестирование ON-LINE
Проверьте свои знания в области open source, ответьте на вопросы в режиме on-line.
Пройти тесты - http://ualinux.com/index.php/ru/online-tests

UA Linux

open source ■ open future

* разработка комплекта дополнений Ubuntu Applications Pack

* разработка, внедрение и сопровождение эффективных IT-решений на базе открытого ПО

* IT-аутсорсинг

* поддержка и консультирование пользователей

* внедрение систем виртуализации

* тестирование оборудования, а также доработка под него программного обеспечения

Внимание!!!

С приходом лета, приходят хорошие новости!

Мы объявляем подписку на печатный журнал UserAndLINUX!

Следите за анонсами на странице журнала http://ualinux.com/index.php/journal

От редакции

Дорогие читатели!

Редакция журнала UserAndLINUX с радостью представляет вашему вниманию новый выпуск приложения Больше чем User.

Новый номер - новые интересные статьи-новая информация. И в этом номере вы узнаете об установке и настройке Ubuntu Server 10.04 LTS в инструкции по серверу Ejabberd от Олега Деордиева; с URPMI вас познакомит Андрей Кондратьев в кратком руководстве пользователя.

В рубрике Server речь пойдет об особенностях настройки PPPoE с шейпингом трафика для небольшой сети.

Необходимый минимум знаний по Vim вы получите в рубрике Console.

Управление несколькими компьютерами одной клавиатурой и мышью, Laptop Mode Tools – утилита для уменьшения энергопотребления ноутбука, все это и ещё много интересного вы сможете прочитать в приложении Больше чем User.

От номера к номеру мы подбираем все более полезную и познавательную информацию, примеры кода и просто интересные советы. Мы надеемся, что приложение Больше чем User станет вашим путеводителем, поможет вам в освоении мира СПО.

Ирина Сикач
Главный редактор приложения
Больше чем User

НАША КОМАНДА:

Руководитель проекта:
Алексей Невенчаный

Главный редактор:
Ирина Сикач

Редакторы:
Владимир Попов
Надежда Козаченко

Дизайн и верстка:
Руслан Гордиевич
Игорь Шарай
Александр Никитин

Подбор материала:
Владимир Попов
Ирина Сикач

Адрес журнала в интернете:
http://ualinux.com/index.php/journal

Обсуждение журнала в форуме:
http://ualinux.com/index.php/forum

По вопросам
преобретения журнала:
http://ualinux.com/index.php/get-journal

Адрес редакции:
Украина, 03040, г.Киев,а/я 56
Email: journal@ualinux.com

Тип издания:
электронный/печатный
Регулярность: ежемесячный
Дата выпуска: 14.03.2011
Тираж: *более10 000 копий.

* указано сумарное количество загрузок прошлого выпуска журнала с первичных источников, а также загрузок с других известных ftp, http и torrent серверов

Все права на материал принадлежат их авторам и опубликованы в открытых источниках.
Адреса на оригинальные источники публикуются.

UA Linux
open source ■ open future

Содержание

Servers

Hi-tech

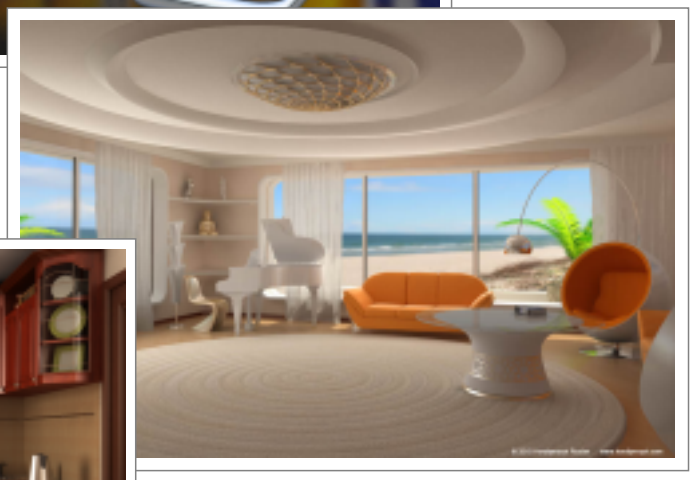
Console

Programming

Other

Ruslan Hordiyevych

дизайн-студия Руслана Гордиевича



дизайн

3D-моделирование

визуализация

www.hordiyevych.com

Настройка PPPoE с шейпингом трафика для небольшой сети

Однажды возникла задача настроить раздачу интернета на пару десятков компьютеров (офисные и домашние). Коробочные решения оказались либо платными, либо достаточно сложно настраиваемыми, поэтому было принято решение использовать собственное на базе Debian Linux. Опытом его поднятия я хочу поделиться с вами в этом посте, но приведенные здесь версии могли немного устареть с момента написания мануала «для себя», так что нужно проявить некоторую внимательность. Также, нужно учитывать, что приведенное решение далеко от идеального и профессионального. Оно поможет скорее тем, кому нужно быстро поднять у себя сервер, раздающий интернет. В конце у нас будет раздача интернета через PPPoE с назначением внутренних IP клиентам, шейпинг трафика, DNS сервер и простой мониторинг текущих сессий из консоли.

ПОДГОТОВКА СИСТЕМЫ

```
aptitude update
aptitude install openssh-server mc
mtr tracert nano tcpdump bind9 pppoe
freeradius radiusclient1 rcconf php5
```

НАСТРОЙКА DNS-СЕРВЕРА

Выполняем

```
nano /etc/bind/named.conf.options
```

Добавляем в конец файла:

```
allow-transfer { none; };
allow-query { 10.128.0.0/16; localhost;
}; //подсеть наших пользователей
allow-recursion { 10.128.0.0/16;
localhost; }; //рекурсия только локальным
version «My ISP DNS»; //скрыли информа-
цию о версии сервера
```

Начнем настраивать chroot-окружение для нашего сервера DNS. Создадим каталоги корневой файловой системы сервера DNS — root-dns:

```
mkdir -p /root-dns/etc
mkdir /root-dns/dev
mkdir -p /root-dns/var/cache/bind
mkdir -p /root-dns/var/run/bind/run
/etc/init.d/bind9 stop
nano /etc/default/bind9
```

Нашли строчку «OPTIONS=» и заменили:

```
OPTIONS=»-u bind -t /root-dns»

mv /etc/bind /root-dns/etc
ln -s /root-dns/etc/bind /etc/bind
nano /etc/init.d/bind9
```

Запускаем:

```
nano /etc/init.d/rsyslogd
```

Нашли «RSYSLOGD_OPTIONS» и заменили:

```
RSYSLOGD_OPTIONS=»-c3 -a /root-dns/dev/
log»
```

Защитим от редактирования и удаления файл конфигурации named.conf:

```
chattr +i /root-dns/etc/bind/named.conf
```

Примечание. Не забудьте снять атрибут «i» перед редактированием файла конфигурации (chattr -i …)

```
mknod /root-dns/dev/null c 1 3
mknod /root-dns/dev/random c 1 8
chmod 666 /root-dns/dev/null /root-dns/
dev/random
chown -R bind:bind /root-dns/var/*
chown -R bind:bind /root-dns/etc/bind
/etc/init.d/bind9 start
```

СЕРВЕР PPPOE

```
cd /tmp
apt-get build-dep pppoe
apt-get source pppoe
cd rp-pppoe-3.8/src
./configure
```

Запускаем:

```
nano config.h
```

Теперь, в только что созданном файле config.h, нужно заменить строчку:

```
/* #undef HAVE_LINUX_KERNEL_PPPOE */
```

на строчку:

```
#define HAVE_LINUX_KERNEL_PPPOE 1
```

Выполняем:

```
cd ..
./debian/rules PLUGIN_PATH=/usr/lib/
pppd/2.4.4/rp-pppoe.so
./debian/rules binary
cd ..
dpkg -i pppoe_3.8-3_i386.deb
```

Заморозим пакет в системе, чтобы предотвратить его автоматическое обновление:

```
echo pppoe hold | dpkg –set-selections
```

С этого момента нужно, как минимум, при каждом обновлении системы просматривать список обновлений. Если для pppoe будет выпущено обновление, закрывающее уязвимость, необходимо будет собрать пакет из свежих исходников заново, снять фиксацию пакета в системе с помощью команды:

```
echo pppoe install | dpkg –set-selections
```

Теперь в системе установлен пакет с PPPoE-сервером, позволяющим поддерживать PPPoE-соединения на уровне ядра. Для запуска сервера в этом режиме, к команде, описанной в предыдущем разделе, нужно добавить опцию -k:

```
pppoe-server -I eth1 -L 192.168.0.1 -k
```

Запускаем:

```
nano /etc/ppp/pppoe-server-options
```

Вставляем:

```
logfile /var/log/pppoe.log
debug
mtu 1472
mru 1472
auth
require-pap
#require-chap
default-asyncmap
ktune
lcp-echo-interval 20
lcp-echo-failure 2
ms-dns 10.128.0.1
plugin radius.so
plugin radattr.so
10.128.0.1:
nobsdcomp
noccp
noendpoint
noipdefault
noipx
novj
receive-all
```

Запускаем:

```
nano /etc/ppp/options
```

В конфиге /etc/ppp/options: убираем всё, ставим:

```
lock
```

Запускаем:

```
nano /etc/init.d/pppoe-server
```

Скрипт:

```
#!/bin/bash
# init file for rp-pppoe server
#
# description: PPPOE kernel mode server
#
# processname: pppoe-server
# chkconfig: - 45 45
# source function library
#. /etc/rc.status
case «$1» in
start)
echo -n «Starting PPPOE server: «
/usr/sbin/pppoe-server -I eth0 -L
10.128.0.1 -R 10.128.1.1 -k
#Здесь eth0 – интерфейс, который смотрит
в локальную сеть
#10.128.0.1 - IP PPPoE сервера
#10.128.1.1 - Первый IP адрес клиента
#touch /var/lock/subsys/pppoed
#rc_status -v
;;
stop)
echo -n «Shutting down PPPOE server: «
pkill pppoe-server
#rm -f /var/lock/subsys/pppoed
#rc_status -v
;;
restart)
$0 stop
$0 start
                                                ;;
status)
status pppoe-server
;;
*)
echo «Usage: pppoed
{start|stop|restart|status}»
exit 1
esac
exit 0
```

Запускаем:

```
rcconf
```

Отключаем bind. Потом туда опять заходим и видим, что он вообще исчез из списка. Внизу он всё-таки есть, а рядом наш pppoe-server. Включили и то, и то. Вышли.

```
cd /root-dns/etc/bind
rndc-confgen -r /dev/urandom -a
chgrp bind rndc.key
chmod +r rndc.key
nano named.conf.options
```

В конце файла добавляем:

```
controls {
inet 127.0.0.1 allow { localhost; } keys
{ rndc-key; };
};
include «/etc/bind/rndc.key»;
```

Теперь ещё нужно включить форвардинг в ядре и сделать подхват правил iptables. Конфигурация пока будет тестовой.

ПАРАМЕТРЫ SYSCTL.CONF, IPTABLES, НАСТРОЙКИ СЕТЕВЫХ ИНТЕРФЕЙСОВ

Запускаем:

```
nano /etc/sysctl.conf
```

Раскомментировали, добавили чего небыло:

```
net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.all.rp_filter=1
net.ipv4.tcp_syncookies=1
net.ipv4.ip_forward=1
kernel.panic = 1
kernel.panic_on_oops = 1
kernel.panic_on_io_nmi = 1
kernel.panic_on_unrecovered_nmi = 1
```

Последние 4 строчки — это перезагрузка при различных падениях системы, чтобы не ездить ради перезагрузки куда-то в другой город. Сохранили, далее делаем тестовый конфиг для iptables:

```
iptables -t nat -F
iptables-save > /etc/firewall.conf
nano rc.local
```

Теперь в rc.local у нас будет:

```
ifconfig eth0 up
iptables-restore </etc/firewall.conf
```

Далее, конфигурируем /etc/network/interfaces — тут eth0 получился локальным, для него ничего вообще не пишем, разве что auto eth0. А по поводу eth1 — если это интернет — я бы задумался о статическом внешнем айпи. В противном случае, нельзя использовать SNAT, а только MASQUERADE, а он сильнее грузит систему.

Запускаем:

```
nano /etc/network/interfaces
```

Конфиг:

```
auto lo
iface lo inet loopback
auto eth0
auto eth1

allow-hotplug eth0 eth1
iface eth1 inet dhcp
```

В данном листинге eth1 — интернет через NAT и DHCP конфиг.

Пробный запуск сервера PPPoE

Далее надо протестировать PPPoE без радиуса. Для этого комментируем 2 строчки плагинов в /etc/ppp/pppoe-server-options.

Запускаем:

```
nano /etc/ppp/pppoe-server-options
```

Конфиг:

```
#plugin radius.so
#plugin radattr.so
```

Запускаем:

```
nano /etc/ppp/pap-secrets
```

Дописываем в конец либо заменяем полностью содержимое:

```
test * test 10.128.2.10
```

Запускаем pppoe-server:

```
/etc/init.d/pppoe-server start
```

Перезагрузили сервер, проверили PPPoE и инет. На любом клиенте создаем PPPoE соединение и подключаемся, используя логин и пароль test. Для того, чтобы инет работал у клиента, нужно включить маскарадинг:

```
iptables -t nat -A POSTROUTING -o eth1
-j MASQUERADE
```

Работает? Хорошо, вернули, как было:

```
iptables -t nat -F
```

FreeRadius

Запускаем:

```
nano /etc/radiusclient/servers
```

Заменяем содержимое файла:

```
127.0.0.1 my-isp-radius
```

Запускаем:

```
nano /etc/radiusclient/radiusclient.conf
```

Заменяем содержимое файла:

```
auth_order radius,local
login_tries 4
login_timeout 60
nologin /etc/nologin
issue /etc/radiusclient/issue
# RADIUS settings
authserver localhost:1812
acctserver localhost:1812
servers /etc/radiusclient/servers
dictionary /etc/radiusclient/dictionary
login_radius /usr/sbin/login.radius
# RADIUS server
seqfile /var/run/radius.seq
mapfile /etc/radiusclient/port-id-map
default_realm
radius_timeout 10
radius_retries 3
login_local /bin/login
```

Запускаем:

```
nano /etc/freeradius/radiusd.conf
```

Заменяем содержимое файла:

```
prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = /var/log/freeradius
raddbdir = /etc/freeradius
radacctdir = ${logdir}/radacct
# Месторасположение конфигурационных и
лог файлов.
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/
freeradius
```

```
  # Каталог с подгружаемыми модулями.
  libdir = ${exec_prefix}/lib
  # Месторасположение pid-файла. Содержащего идентификатор процесса.
  pidfile = ${run_dir}/freeradius.pid
  # Имя пользователя и группа от которых запускается FreeRADIUS
  user = freerad
  group = freerad
  # Максимальное время (в секундах) используемое для обработки запроса.
  max_request_time = 30
  # Удалить запросы которые обрабатываются более чем max_request_time
  delete_blocked_requests = no
  # Время ожидания (в секундах) перед очисткой reply запроса отправленного NAS.
  cleanup_delay = 5
  # Максимальное количество запросов хранимых сервером. Это число должно быть равно
  # количеству клиентов помноженному на 256.
  # К примеру для четырех клиентов оно будет 1024.
  max_requests = 5120
  # Закрепить за ip адресом. По умолчанию RADIUS сервер при старте принимает
  # запросы со всех ip адресов.
  bind_address = 127.0.0.1
  # Закрепить за FreeRADIUS конкретный port. Если указан ноль,
  # то значение берется из /etc/services
  port = 0
  # Запретить/разрешить ip адреса в dns имена.
  # Включение этой опции может сильно снизить производительность.
  hostname_lookups = no
  # Создавать/несоздавать отладочные файлы при падении сервера.
  allow_core_dumps = no
  # Разрешить использование регулярных выражений.
  regular_expressions = yes
  extended_expressions = yes
  # Записывать полный User-Name аттрибут если найден в запросе.
  log_stripped_names = no
  # Записывать в лог попытки авторизации.
  log_auth = yes
  # Записывать в логи пароли при авторизации.
  # log_auth_badpass - не корректные пароли
  # log_auth_goodpass - корректные пароли
  log_auth_badpass = yes
  log_auth_goodpass = no
  # Включить/выключить коллизию пользователей.
  usercollide = no
  # конвертировать логин и/или пароль до или после авторизации.
  lower_user = no
  lower_pass = no
  # удалить пробелы в логине и/или пароле.
  nospace_user = no
  nospace_pass = no
  # настройки безопасности от возможных DoS аттак.
  Security {
  # Максимальное допустимое количество аттрибутов в RADIUS пакете.
  max_attributes = 200
  # Задержка (в секундах) перед отправкой Access-Reject пакета.
  reject_delay = 1
  # Не отвечать на запросы Status-Server
  status_server = no
  }
  # Конфигурация клиентов RADIUS сервера.
  # Описывается в отдельном файле.
  $INCLUDE ${confdir}/clients.conf
  # Отключить snmp поддержку.
  snmp=no
  # Настрока пула процессов.
  thread pool {
  # количество первоначально запущенных процессов.
  start_servers = 5
  # Максимально возможное количество процессов.
  max_servers = 32
  # Динамическая регулировка количества процессов.
  min_spare_servers = 3
  max_spare_servers = 10

  # Количество принимаемых запросов процессом. МОжет помочь при утечках памяти в
  # RADIUS сервере. Если выставить 300, процессы будут периодически перегружаться
  # для уборки мусора.
  max_requests_per_server = 0
  }

  # Секция конфигурации динамических модулей.
  Modules {
  # Модуль PAP авторизации.
  # Необходим для обработки запросов с PAP авторизацией.
  # encryption_scheme указывает в каком виде хранятся пароли.
  # clear - подразумевает в открытом виде.
  Pap {
  encryption_scheme = clear
  }
  # Модуль CHAP авторизации.
  # Необходим для обработки запросов с CHAP авторизацией.
  # authtype подразумевает обработку за-
```

```
просов только с аттрибутом Auth-Type=CHAP
  chap {
  authtype = CHAP
  }
  # Модуль преобработки запросов.
  # Т.е. перед авторизацией пакета.
  Preprocess {
  # huntgroups - хинт группы см. файл
huntgoups.
  # hints – хинты.
  huntgroups = ${confdir}/huntgroups
  hints = ${confdir}/hints

  # Обработка Cisco VSA.
  with_cisco_vsa_hack = no
  }
  # Модуль Microsoft CHAP авторизации.
  # Поддерживает так же еще и Microsoft
CHAP v2
  # authtype подразумевает обработку за-
просов только с аттрибутом Auth-Type=MS-
CHAP
  # use_mppe = no указывает на отсутствие
компресии VPN туннеля.
  Mschap {
  authtype = MS-CHAP
  use_mppe = no
  }
  # Модуль записей Livingston RADIUS типа.
  # usersfile содержит авторизационные за-
писи пользователей.
  # Рекомендуется использовать только для
тестов и выставления значений по умолча-
нию.
  # acctusersfile содержит пользователей
подлежащих учету (аккаунтингу).
  # compat - совместимость. При использо-
вании файлов только FreeRADIUS можно от-
ключить.
  Files {
  usersfile = ${confdir}/users
  compat = no
  }

  # Запись детального лога аккаунтинговых
пакетов.
  Detail {
  detailfile = ${radacctdir}/%{Client-IP-
Address}/detail-%Y%m%d
  detailperm = 0600
  }

  # Запись детального лога пакетов автори-
зации.
  detail auth_log {
  detailfile = ${radacctdir}/%{Client-IP-
Address}/auth-detail-%Y%m%d
  detailperm = 0600
  }

  # Запись детального лога reply пакетов.
  detail reply_log {
  detailfile = ${radacctdir}/%{Client-IP-
Address}/reply-detail-%Y%m%d
  detailperm = 0600
  }

  # Создать уникальный ключ для аккаунтинг
сессии.
  # Многие NAS повторно используют Acct-
Session-ID.
  # key перечисление аттрибутов для гене-
рации Acct-Session-ID
  acct_unique {
  key = «User-Name, Acct-Session-Id, NAS-
IP-Address, Client-IP-Address, NAS-Port-
Id»
  }

  # Конфигурация авторизации и аккаунтинга
посредством СУБД
  # содержится в отдельном файле cakesql.
conf
  #$INCLUDE ${confdir}/sql.conf
  #$INCLUDE ${confdir}/sql/mysql/counter.
conf
  }

  # Авторизация
  # сначала идет пакет передается в
preprocess
  # где может быть модифицирован.
  # Далее chap mschap обрабатывают chap и
mschap авторизацию.
  Authorize {
  preprocess
  chap
  mschap
  # Не ведем логи пакетов авторизации.
  # auth_log
  files
  #sql
  }
  # Аунтификация
  # Секция содержит модули доступные, для
аунтификации.
  Authenticate {
  Auth-Type PAP {
  pap
  }

  Auth-Type CHAP {
  chap
  }

  Auth-Type MS-CHAP {
  mschap
  }

  }

  # Преобразование аккаунтинговых пакетов.
```

```
  Preacct {
  preprocess
  }
  # Секция ведения аккаунтинга.
  Accounting {
  # Создание Acct-Session-Id если ваш NAS
генрит их вполне корректно можете убрать.
  acct_unique
  # Не создаем detail лог.
  detail
  # Помещать аккаунтинговые пакеты в СУБД
  #sql
  }
  # Секция ведения логов reply-пакетов.
  Post-auth {
  # Не ведем детальный лог репли пакетов.
  reply_log
  }
  log {
  destination = files
  file = ${logdir}/radius.log
  syslog_facility = daemon
  stripped_names = no
  auth = no
  auth_badpass = no
  auth_goodpass = no
  }
```

Запускаем:

```
nano /etc/freeradius/clients.conf
```

Заменяем содержимое файла:

```
client localhost {
ipaddr = 127.0.0.1
secret = my-isp-radius
require_message_authenticator = no
nastype = other
}
```

Запускаем:

```
nano /etc/freeradius/users
```

Конфиг:

```
steve Cleartext-Password := «testing»
Service-Type = Framed-User,
Framed-Protocol = PPP,
Framed-IP-Address = 10.128.13.3,
Framed-IP-Netmask = 255.255.0.0,
Framed-Routing = Broadcast-Listen,
Framed-Filter-Id = «std.ppp»,
Framed-MTU = 1500,
Framed-Compression = Van-Jacobsen-TCP-IP

# тут остальные юзеры по такому же прин-
ципу

DEFAULT Framed-Protocol == PPP
Framed-Protocol = PPP,
Framed-Compression = Van-Jacobson-TCP-IP

DEFAULT Hint == «CSLIP»
Framed-Protocol = SLIP,
Framed-Compression = Van-Jacobson-TCP-IP

DEFAULT Hint == «SLIP»
Framed-Protocol = SLIP
```

Вообще в будущем вам нужно будет настроить связку с MySQL, но на данном этапе это не критично. При добавлении пользователя нужно перезапустить Freeradius:

```
/etc/init.d/freeradius restart
```

Теперь нужно разобраться с шейпингом трафика. Идея такова, что скорость должна регулироваться индивидуально для каждого пользователя и передаваться в виде атрибута от Radius-сервера серверу NAT. Зачем? Для того, чтобы в последствии было возможно расширить структуру и распределить нагрузку между несколькими NAS.

ШЕЙПИНГ ТРАФИКА

Итак, подготовка. Установим Wondershaper:

```
aptitude install wondershaper
```

Этот пакет представляет собой один скрипт, работающий с CBQ/HTB приоритезацией и имеющий вменяемый формат вызова. При желании можно протестировать его работу до внедрения:

```
wondershaper ppp0 512 512
```

Естественно, ppp0 должен существовать и быть результатом соединения клиента с нашим сервером. После того, как команда выполнена, можно веб-браузером клиента зайти на любой сайт, измеряющий скорость, типа speedtest.net — и убедиться, что она таки режется шейпером. Для сброса настроек без рассоединения с клиентом (кика то есть), делаем

```
wondershaper clear ppp0
```

и убеждаемся в том, что скорость опять нормальная (то есть равна скорости, которую мы берем от «старшего брата» на WAN интерфейсе). Добавление ограничений скорости нужно автоматизировать, используя простой парсер файлов /var/run/radattr.ppp*, которые появляются при успешном ppp соединении и содержат все атрибуты, передаваемые Radius-сервером серверу PPPoE. Займемся атрибутами. Дополняем содержимое файла дополнительных атрибутов /etc/freeradius/dictionary вот этим:

Запускаем:

```
nano /etc/freeradius/dictionary
```

Конфиг:

```
# Limit session traffic
ATTRIBUTE Session-Octets-Limit 227
integer
# What to assume as limit - 0 in+out, 1
in, 2 out, 3 max(in,out)
ATTRIBUTE Octets-Direction 228 integer
# Connection Speed Limit
```

```
  ATTRIBUTE PPPD-Upstream-Speed-Limit 230
integer
  ATTRIBUTE PPPD-Downstream-Speed-Limit
231 integer
  ATTRIBUTE PPPD-Upstream-Speed-Limit-1
232 integer
  ATTRIBUTE PPPD-Downstream-Speed-Limit-1
233 integer
  ATTRIBUTE PPPD-Upstream-Speed-Limit-2
234 integer
  ATTRIBUTE PPPD-Downstream-Speed-Limit-2
235 integer
  ATTRIBUTE PPPD-Upstream-Speed-Limit-3
236 integer
  ATTRIBUTE PPPD-Downstream-Speed-Limit-3
237 integer
  ATTRIBUTE Acct-Interim-Interval 85
integer
```

Далее, эти же атрибуты нужно добавить в словарь Raduisclient-a. Редактируем /etc/radiusclient/dictionary, в конце добавляем то же самое:

Запускаем:

```
nano /etc/radiusclient/dictionary
```

Конфиг:

```
  # Limit session traffic
  ATTRIBUTE Session-Octets-Limit 227
integer
  # What to assume as limit - 0 in+out, 1
in, 2 out, 3 max(in,out)
  ATTRIBUTE Octets-Direction 228 integer
  …..................................
.........................
  ATTRIBUTE PPPD-Downstream-Speed-Limit-3
237 integer
  ATTRIBUTE Acct-Interim-Interval 85
integer
```

Всё отлично, теперь они (FreeRadius и RadiusClient) друг друга поймут. Соответственно, в настройках пользователей freeradius-a появятся дополнительные строчки с указанием лимита входящей и исходящей скорости (но не обязательные). Пример /etc/freeradius/users с учетом изменений:

Запускаем:

```
nano /etc/freeradius/users
```

Конфиг:

```
steve Cleartext-Password := «testing»
Service-Type = Framed-User,
Framed-Protocol = PPP,
Framed-IP-Address = 10.128.13.3,
Framed-IP-Netmask = 255.255.0.0,
Framed-Routing = Broadcast-Listen,
Framed-Filter-Id = «std.ppp»,
Framed-MTU = 1500,
Framed-Compression = Van-Jacobsen-TCP-IP
PPPD-Downstream-Speed-Limit = 1024,
PPPD-Upstream-Speed-Limit = 1024

# тут остальные юзеры по такому же принципу
DEFAULT Framed-Protocol == PPP
Framed-Protocol = PPP,
Framed-Compression = Van-Jacobson-TCP-IP

DEFAULT Hint == «CSLIP»
Framed-Protocol = SLIP,
Framed-Compression = Van-Jacobson-TCP-IP

DEFAULT Hint == «SLIP»
Framed-Protocol = SLIP
```

Теперь NAS получает сведения о скорости, но не умеет их обрабатывать. Делаем два скрипта, выполняющих парсинг переданных атрибутов при установлении соединения:

Запускаем:

```
nano /etc/ppp/ip-up.d/0001shaper
```

Скрипт:

```
#!/bin/sh
/root-scripts/shape-on $1 $5
exit 0
```

Запускаем:

```
nano /etc/ppp/ip-down.d/0001shaper
```

Скрипт:

```
#!/bin/sh
/root-scripts/shape-off $1 $5
exit 0
```

Как видно из этих листингов, вызывается некий скрипт, применяющий и отменяющий шейпинг. С этого момента – поподробнее. $1 – это переданное имя интерфейса, то есть ppp0 к примеру, $5 – это Framed-IP, то есть IP клиента PPP. Создадим заветный каталог и положим в него два файла:

```
mkdir /root-scripts
nano /root-scripts/shape-on
```

Скрипт:

```
#!/bin/sh
if [ -f /var/run/radattr.$1 ]
then
DOWNSPEED=`awk '/PPPD-Downstream-Speed-
Limit/ {print $2}' /var/run/radattr.$1`
UPSPEED=`awk '/PPPD-Upstream-Speed-
Limit/ {print $2}' /var/run/radattr.$1`
wondershaper $1 $DOWNSPEED $UPSPEED
fi
iptables -A FORWARD -s $2 -j ACCEPT
iptables -A INPUT -s $2 -j ACCEPT
iptables -t nat -A POSTROUTING -s $2 -o
eth1 -j MASQUERADE
```

Запускаем:

```
nano /root-scripts/shape-off
```

Скрипт:

```
#!/bin/sh
iptables -D FORWARD -s $2 -j ACCEPT
iptables -D INPUT -s $2 -j ACCEPT
iptables -t nat -D POSTROUTING -s $2 -o
eth1 -j MASQUERADE
```

Запускаем:

```
chmod +x /root-scripts/*
```

В листингах eth1 — это всё ещё WAN интерфейс. Итак, что мы получаем в результате: Клиент соединяется с сервером PPP, получает IP адрес. Автоматически вызывается скрипт, настраивающий шейпинг на клиентском PPP интерфейсе и включающий для него маскарадинг. При рассоединении правило маскарадинга удаляется также автоматически. Вроде как wondershaper не нужно принудительно отцеплять от pppx, во всяком случае, мануалы об этом молчат. И всё нормально работает на практике.

ПРОСТАЯ ЗАЩИТА НА ОСНОВЕ IPTABLES

Теперь уделим немножко внимания iptables и безопасности сервера. Заменим /etc/firewall.conf:

Запускаем:

```
nano /etc/firewall.conf
```

Скрипт:

```
#!/bin/bash
#reset all
iptables -F
iptables -Z
iptables -t nat -F
iptables -t nat -Z

#iface config
INET_IFACE=»eth1»
LO_IFACE=»lo»
LO_IP=»127.0.0.1»
#kernel modules
modprobe ip_tables
modprobe ip_conntrack
modprobe iptable_filter
modprobe iptable_mangle
modprobe iptable_nat
modprobe ipt_LOG
modprobe ipt_limit
modprobe ipt_state
modprobe ipt_MASQUERADE
modprobe ip_conntrack_ftp
modprobe ip_conntrack_irc
modprobe ip_nat_ftp
modprobe ip_nat_irc

#default policy
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

#user chains
iptables -N bad_tcp_packets
iptables -N allowed
iptables -N tcp_packets
iptables -N udp_packets
iptables -N icmp_packets
iptables -N blocksshd
blocksshd –start

###bad_tcp_packets chain
iptables -A bad_tcp_packets -p tcp
--tcp-flags SYN,ACK SYN,ACK -m state
--state NEW -j REJECT --reject-with tcp-
reset
#iptables -A bad_tcp_packets -p tcp !
--syn -m state --state NEW -j LOG --log-
prefix «New not syn:»
iptables -A bad_tcp_packets -p tcp !
--syn -m state --state NEW -j DROP

###allowed chain
iptables -A allowed -p TCP --syn -j
ACCEPT
iptables -A allowed -p TCP -m state
--state ESTABLISHED,RELATED -j ACCEPT
iptables -A allowed -p TCP -j DROP

###tcp_packets chain
iptables -A tcp_packets -p TCP -s 0/0
--dport 22 -j blocksshd
iptables -A tcp_packets -p TCP -s 0/0
--dport 22 -j allowed
iptables -A tcp_packets -p TCP -s 0/0
--dport 25 -j allowed
#iptables -A tcp_packets -p TCP -s 0/0
--dport 80 -j allowed

###udp_packets chain
iptables -A udp_packets -p UDP -s 0/0
--destination-port 53 -j ACCEPT

###icmp chain
iptables -A icmp_packets -p ICMP -s 0/0
--icmp-type 8 -j ACCEPT
iptables -A icmp_packets -p ICMP -s 0/0
--icmp-type 11 -j ACCEPT

###input chain
iptables -A INPUT -p TCP -j bad_tcp_
packets
iptables -A INPUT -p ALL -i $LO_IFACE -j
ACCEPT
iptables -A INPUT -p ALL -m state
--state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p TCP -i $INET_IFACE
-j tcp_packets
iptables -A INPUT -p UDP -i $INET_IFACE
-j udp_packets
iptables -A INPUT -p ICMP -i $INET_IFACE
-j icmp_packets
#iptables -A INPUT -m limit --limit 3/
minute --limit-burst 3 -j LOG --log-level
```

```
DEBUG --log-prefix «IPT INPUT packet died:
«

  ###forward chain
  iptables -A FORWARD -p tcp -j bad_tcp_
packets
  #custom rules
  #iptables -A FORWARD -s 10.128.0.10/32
-d 192.168.1.1/32 -p tcp -j ACCEPT
  iptables -A FORWARD -s 10.128.0.0/16 -d
192.168.1.1/32 -p tcp -j DROP
  #/custom rules
  iptables -A FORWARD -m state --state
ESTABLISHED,RELATED -j ACCEPT
  #iptables -A FORWARD -m limit --limit 3/
minute --limit-burst 3 -j LOG --log-level
DEBUG --log-prefix «IPT FORWARD packet
died: «

  ###OUTPUT chain
  iptables -A OUTPUT -p tcp -j bad_tcp_
packets
  iptables -A OUTPUT -p ALL -j ACCEPT
  #iptables -A OUTPUT -m limit --limit 3/
minute --limit-burst 3 -j LOG --log-level
DEBUG --log-prefix «IPT OUTPUT packet died:
«

  ###mangle
  iptables -t mangle -A POSTROUTING -o
$INET_IFACE -j TTL --ttl-set 64
```

Что мы тут видим? Во-первых, выделенное правило (выделено подчеркиванием) в реальной конфигурации или просто отсутствовать за ненадобностью, но смысл таков: если шлюзом для сервера является ADSL модем или подобный девайс с веб-админкой, нам не нужно выдумывать хитрые пароли для входа туда. Вместо этого, мы запрещаем к ним обращаться из клиентских туннелей, но одному таки разрешаем — это будет сервисная учетная запись, которой можно на него ломиться. Само собой, она не для клиентов. Далее. Добавлены команды, включающие защиту от брутфорсинга SSH сервера. Для их работы нужно настроить blocksshd.

BLOCKSSHD

Для начала его нужно скачать с blocksshd.sourceforge.net/ и разархивировать файлы куда-нибудь. Я использовал для закачки wget, но путь — понятное дело — вырезал со странички SourceForge. Тоесть он будет, скорее всего, другим:

```
  cd /tmp
  wget downloads.sourceforge.net/project/
blocksshd/blocksshd/BlockSSHD%201.3/
blocksshd-1.3.tar.gz?use_mirror=kent
```

Этот скрипт написан на Perl и требует некоторых дополнительных модулей для работы. Итак, займемся модулями:

```
  cpan
  upgrade
  install Sys::Syslog
  install Sys::Hostname
  install File::Tail
  install Tie::File
  install Net::DNS
  install Net::Subnets
  install Getopt::Long
  exit
```

Обращаю внимание на то, что всё это одним махом не нужно копировать и вставлять в терминал. Это потому, что каждая операция требует большого количества времени для выполнения и в процессе всей установки Cpan будет задавать вопросы. На все вопросы можно просто нажимать Enter, там по умолчанию всё правильно, как ни странно.

Теперь нужно распаковать blocksshd и установить его:

```
  cd /tmp
  tar -xvzf ./blocksshd-1.3.tar.gz
  cd blocksshd-1.3
  make install
```

Ну и естественно, версия на момент написания мануала была именно эта, а может быть любая, так что нужно внимательно всё сделать, а не копипастить в консоль. Осталось отредактировать конфигурацию blocksshd:

Запускаем:

```
  nano /etc/blocksshd.conf
```

Конфиг:

```
  $cfg = {
  os => 'linux', # Target OS - either
linux or bsd
  chain => 'blocksshd', # Name of iptables
or pf chain
  logfile => '/var/log/secure', # Log file
to monitor
  logcheck => '10', # How often to check
the log file
  max_attempts => '4', # Max number of
failures
  timeout => '360', # Reset IP count if no
activity after time out in seconds
  unblock => '1', # Enable unblocking
  unblock_timeout => '43200', # Time in
seconds after which to unblock a blocked
IP address
  restore_blocked => '0', # Turn on
checking for previously blocked Ips
  log_ips => '/etc/blocksshd.list', # Log
file for blocked Ips
  pid_file => '/var/run/blocksshd.pid', #
Location of PID file
  send_email => '0', # Enable the sending
of email notifications
  email => 'root', # Email address to send
notifications
  mail => '/bin/mail', # Location of mail
binary
```

```
  email_whois_lookup => '1', # enable
whois lookup of the blocked ip addres in
the sent email
  whois => '/usr/bin/whois', # location of
the whois binary
  sed => '/bin/sed', # location of the sed
binary
  iptables => '/sbin/iptables', # Location
of iptables binary - only for Linux
  pfctl => '/sbin/pfctl', # Location of
pfctl binary - only for BSD
  whitelist => [qw{
  127.0.0.1/32
  }], # whitelist - list of IPs that will
never be blocked - IPs must be specified in
the form ad$
  };

  #leave 1; here!
  1;
```

В этом листинге нужно обратить внимание на выделенные строчки. Первая — максимальное число попыток входа с одного адреса, вторая — нужно ли разблокировать забаненный айпи по истечении времени в секундах, указанного в третьей строчке. Четвертая — оповещать ли по email о попытках взлома.

Теперь можно перезагрузить сервер и тестировать то, что получилось.

ПРОСТЕЙШИЙ МОНИТОРИНГ

Для удобства мониторинга активных сессий можно использовать скрипт, написаный «на коленке». Создадим его и настроим симлинк для простого запуска, ну и проверим, конечно:

Запускаем:

```
nano /root-scripts/clients
```

Скрипт:

```
#!/usr/bin/php
<?
function time_since($original) {
$chunks = array(
array(60 * 60 * 24 , 'D '),
array(60 * 60 , 'h:'),
array(60 , 'min'),
);
$today = time();
$since = $today - $original;
for ($i = 0, $j = count($chunks); $i <
$j; $i++) {
$seconds = $chunks[$i][0];
$name = $chunks[$i][1];
if (($count = floor($since / $seconds))
!= 0) {
break;
}
}
$print = «$count{$name}»;
if ($i + 1 < $j) {
$seconds2 = $chunks[$i + 1][0];
$name2 = $chunks[$i + 1][1];
if (($count2 = floor(($since - ($seconds
* $count)) / $seconds2)) != 0) {
$print .= «$count2{$name2}»;
}
}
return $print;
}

function execShellCmdRaw($cmd) {
ob_start();
passthru($cmd);
$buffer = ob_get_contents();
ob_end_clean();
return $buffer;
}

function rawToArray($buffer) {
return empty($buffer)?array():preg_
split('/[\r\n]+/', $buffer, -1, PREG_
SPLIT_NO_EMPTY);
}

function execShellCmd($cmd) {
return rawToArray(execShellCmdRaw($c
md));
}

function getNATSources() {
$nat = execShellCmd('iptables -t nat -L
| grep MASQUERADE');
$nats = array();
foreach ($nat as $s) {
$pos_start = strpos($s, '--') + 2;
$pos_end = strpos($s, 'anywhere');
$n = trim(substr($s, $pos_start, $pos_
end - $pos_start));
$nats[$n] = $n;
}
return $nats;
}

function getTunnelMacAddr($pid) {
$pppdc = execShellCmd('ps -p '.$pid.' -f
| grep pppd');
$pppdc = $pppdc[0];
$pos_start = strpos($pppdc, '-e ') + 3;
$pos_end = strpos($pppdc, '-S \'\'');
return trim(substr($pppdc, $pos_start,
$pos_end - $pos_start));
}

function getPPPTunnels() {
$ifc = execShellCmd('ifconfig -a');
$tunnels = array();
$nat = getNATSources();
foreach ($ifc as $i => $s) {
if (substr($s, 0, 3) == 'ppp') {
$ppp = array();
$ppp['interface'] = substr($s, 0, 4);
```

```
   $ppp['framed_ip'] =
trim(substr($ifc[$i+1], strpos($ifc[$i+1],
'P-t-P:') + 6, 12));
   $RXpos_start = strpos($ifc[$i+6], '(') +
1;
   $RXpos_end = strpos($ifc[$i+6], ')');
   $ppp['tx'] = substr($ifc[$i+6], $RXpos_
start, $RXpos_end - $RXpos_start); //
TX(user) = RX(server)
   $tx_sub = substr($ifc[$i+6], $RXpos_end
+ 1);
   $TXpos_start = strpos($tx_sub, '(') + 1;
   $TXpos_end = strpos($tx_sub, ')');
   $ppp['rx'] = substr($tx_sub, $TXpos_
start, $TXpos_end - $Txpos_start);
   $ppp['nat'] = array_key_
exists($ppp['framed_ip'],
$nat)?'on':'off';
   $pidf = '/var/
run/'.$ppp['interface'].'.pid';
   if (file_exists($pidf)) {
   $ppp['uptime'] = time_
since(filemtime($pidf));
   $ppp['pid'] = trim(file_get_
contents($pidf));
   $ppp['mac_addr'] =
getTunnelMacAddr($ppp['pid']);
   $ppp['username'] = 'detectedusername';
   $tunnels[] = $ppp;
   }
   }
   }
   return $tunnels;
   }

   function clientTable($tunnels) {
   printf(«\n»);
   printf(« %15s | «,»IP Address»);
   printf(«%20s | «,»MAC Address «);
   printf(«%3s | «,»NAT»);
   printf(«%16s | «,»User name «);
   printf(«%9s | «,»TX (up)»);
   printf(«%9s | «,»RX (down)»);
   printf(«%11s»,»Uptime «);
   printf(«\n «);
   for ($i = 0; $i < 101; $i++)
printf(«-»);
   printf(«\n»);
   foreach($tunnels as $tun) {
   printf(« %15s | «,$tun['framed_ip']);
   printf(«%20s | «,$tun['mac_addr']);
   printf(«%3s | «,$tun['nat']);
   printf(«%16s | «,$tun['username']);
   printf(«%9s | «,$tun['tx']);
   printf(«%9s | «,$tun['rx']);
   printf(«%11s»,$tun['uptime']);
   printf(«\n»);
   }
   printf(«\n»);
   }

   function addRadParams(&$tunnels = NULL)
{
   $radusers = rawToArray(file_get_
contents('/etc/freeradius/users'));
   $users = array();
   foreach ($radusers as $i => $s) {
   if (strpos($s, 'Cleartext-Password') !==
FALSE) {
   $user = array();
   $user['username'] = trim(substr($s, 0,
strpos($s, 'Cleartext-Password')));
   $si = $i + 1;
   while (strpos($radusers[$si], 'Framed-
IP-Address') == FALSE) $si++;
   $ip_raw = explode('=', $radusers[$si]);
   $ip_raw = $ip_raw[1];
   $users[trim($ip_raw, ' ,')] = $user;
   }
   }
   foreach ($tunnels as $i => $tun) {
   $tunnels[$i]['username'] =
$users[$tun['framed_ip']]['username'];
   }
   return $tunnels;
   }

   $t = getPPPTunnels();
   addRadParams($t);
   clientTable($t);
```

Запускаем:

```
chmod +x /root-scripts/clients
ln -s /root-scripts/clients /usr/bin/
clients
clients
```

Получим что-то типа:

IP Address | MAC Address | NAT | User name | TX (up) | RX (down) | Uptime

---

10.128.1.7 | 1:00:19:66:df:39:26 | on | room56 | 507.4 MiB | 445.5 MiB | 16h:31min
10.128.1.3 | 23:00:13:8f:70:30:03 | on | room47 | 8.2 MiB | 137.8 MiB | 1h:14min
10.128.1.5 | 3:00:a1:b0:11:74:cf | on | room50 | 19.2 MiB | 500.2 MiB | 16h:30min

Вот и всё, наш сервер готов к использованию. По своим наблюдениям могу сказать, что простого системника с 1 гб DDR1 и 2 ГГц процессором хватает для раздачи 16-мегабитного канала для 30 пользователей (больше просто не нужно было) и ощутимой нагрузки система не испытывает.

Удачи начинающим провайдерам!

http://habrahabr.ru

Интсрукция по серверу Ejabberd

Установка и настройка Ubuntu Server 10.04 LTS

РЕСУРСЫ ПО ТЕМЕ

Фан-сайт – www.ejabberd.im
Коммерческий сайт – www.process-one.net
Онлайн гайд – www.process-one.net
Частые вопросы – www.ejabberd.im
DevDoc – www.process-one.net
Модули – www.ejabberd.im
Модуль log_chat – www.svn.process-one.net
WinSCP – www.winscp.net
VacuumIM – www.vacuum-im.org

УСТАНОВКА ОС UBUNTU SERVER 10.04 LTS

1) Вставляем диск с дистрибутивом Linux Ubuntu Server 10.04 LTS x86
2) Выбираем Русский язык
3) Выбираем пункт №1. Установить Ubuntu Server
4) Выбираем регион Украина
5) Выбираем - НЕТ - не определять раскладку клавиатуры
6) Соответственно выбираем вручную США
7) Главное меню тоже США
8) Дальше система сканирует диск с дистрибутивом
9) После загрузки с диска нужных компонентов система спросит имя для этой машины в сети. В нашем случае выбираем имя «Jabber»
10) Дальше говорим, что наша временная зона является не корректной (НЕТ) и вручную выбираем «Kiev»
11) Дальше вручную размечаем диск на усмотрение главное — swp, / , /home
12) Соглашаемся с тем, что данные изменения теперь будут записаны на диск
13) Пошла установка базовой системы
14) Понадобится использование интернета. В данном руководстве подразумевается, что в сети есть DHCP сервер и выход в интернет.
15) Теперь введите имя пользователя и пароль. Этот пользователь имеет право на использование команды "sudo", то есть он является sudouser
16) Вводим имя «administrator» и стандартный пароль
17) Нажимаем «НЕТ» - не использовать шифрование домашней директории
18) Дальше нас спросят о настройках ПРОКСИ в нашей сети, если ничего такого у нас нет, значит просто жмём ввод (enter) или же указываем настройки прокси
19) Теперь опять будет использован интернет
20) На вопрос про способ обновления, говорим - без автоматического обновления.
21) Для обновления вручную, вводим последовательность команд
- sudo -s
- aptitude update
- aptitude upgrade -y

22) Теперь нужно выбрать дополнительное програмное обеспечение. Выбираем только OpenSSH (пробелом помечаем)
23) Снова подключается к интернету и вытягивает нужные компоненты
24) Соглашаемся на установку GRUB Loader в MBR
25) Далее машина выкинет CD с дистрибутивом и скажет: убедитесь, что диск извлечён и продолжайте установку
26) Перезагрузка
27) После перзагрузки нас спросят Login и пароль пользователя

НАСТРОЙКА ОС UBUNTU SERVER 10.04 LTS

1) Вводим «administrator» со стандартным паролем
2) Сразу видно количество обновлений (около 100)
3) Выполняем вручную обновление системы
4) Напоминаю последовательность – "sudo -s" поле этого вводим пароль пользователя администратора и выполняем - "aptitude update && aptitude upgrade -y"
5) Для этих операций необходим интернет
6) Перезагружаем после обновлений команда «reboot»

УСТАНОВКА И НАСТРОЙКА СЕРВЕРА EJABBERD

1) Повышаем права «sudo -s», затем выполняем «aptitude install ejabberd -y»
2)Проинсталировали, теперь поставим нужные программы для удобства
3) Выполняем команду - aptitude install htop mc trafshow traceroute
4) Запускаем mc
5) Переходим в директорию /etc/ejabberd/
6) Находим ejabberd.cfg и на нём жмём клавишу F4
7) И приводим к виду (пример этого файла снизу)

ejabberd.conf

```
  % Символ «%» говорит Erland"у, что данная
строка это комментарий
  % а комментарии пропускаются при чтении па-
раметров из файла конфигурации
  % Имя машины сервера
  {hosts, [«jabber»]}.
  % Уроень логирования
  {loglevel, 4}.
  %% Слушать порты
  % Слушать порт для подключения клиентов к
серверу (c2s)
  {listen,
   [
    {5222, ejabberd_c2s, [
              {access, c2s},
              {shaper, c2s_shaper},
              {max_stanza_size, 65536},
              starttls, {certfile, «/etc/
ejabberd/ejabberd.pem»}
             ]},
    % Слушать Web панель администратора
    {5280, ejabberd_http, [
              http_bind,
              http_poll,
              web_admin
```

```
            ]}
    ]}.

   %% Авторизация
   % Метод авторизации внутренний
   {auth_method, internal}.
   %%  Трафик Шейпера (профили)
   % Создн нормальный шейпер он ограничивает до
1000 B/s
   {shaper, normal, {maxrate, 1000}}.
   % Быстрый Шейпер ограничивает  до 50000 B/s
   {shaper, fast, {maxrate, 50000}}.
   %%%   Лист Контроля Доступа
   % Кто админ
   {acl, admin, {user, «admin», «jabber»}}.
   {acl, admin, {user, «kvin», «jabber»}}.
   % Локальные пользователи (не редактировать
эти строки)
   {acl, local, {user_regexp, «»}}.
   %% Правила доступа
   % Максимально кол-во одновременно открытых
сессий одним клиентом
   {access, max_user_sessions, [{1, all}]}.
   % Иаксимальное кол-во офлайн сообщений
   {access, max_user_offline_messages, [{10,
all}]}.
   % Доступ только локальным пользователям
   {access, local, [{allow, local}]}.
   % Только не заблокированые пользователи мо-
гут соединятся с сервером
   {access, c2s, [{deny, blocked},
              {allow, all}]}.
   % Все кроме админов используют нормальный
шейпер
   {access, c2s_shaper, [{none, admin},
              {normal, all}]}.
   % А вот соединения Сервер к Серверу (s2s)
используют быстрый шейпер
   {access, s2s_shaper, [{fast, all}]}.
   % Только админы могут создавать широковеща-
тельные сообщения
   {access, announce, [{allow, admin}]}.
   % Только админы могут использовать интерфейс
конфигурирования
   {access, configure, [{allow, admin}]}.
   % Админы сервера так же админы и конференций
   {access, muc_admin, [{allow, admin}]}.
   % Всем пользоваетелям разрешено пользоватся
конференциями
   {access, muc, [{allow, all}]}.

   % Кто угодно может создавать конференции
   {access, pubsub_createnode, [{allow, all}]}.

   % Опредеяет параметры сервера
   {host_config, «jabber»,
    [
    % Разрешить соединения c2s админам и всем
остальным
     {access, c2s, [{allow, admin}, {allow,
all}]},
   % Самовольная регистрация на сервере запре-
щена
     {access, register, [{deny, all}]}
    ]
   }.
   %%   Языки по умолчанию

   % Сообщения сервера
   {language, «ru»}.
   % Для каждого виртуального сервера опреде-
лить язык
    {host_config, «jabber»,
     [{language, «ru»}]
    }.
   %%   Модули

   {modules,
    [
     {mod_adhoc,     []},
     {mod_announce, [{access, announce}]}, %
requires mod_adhoc
     {mod_caps,      []},
     {mod_configure,[]}, % requires mod_adhoc
     {mod_admin_extra, []},
     {mod_disco,     []},
     %%{mod_echo,    [{host, «echo.
localhost»}]},
     {mod_irc,       []},
     {mod_last,      []},
     {mod_muc,       [
          {access, muc},
          {access_create, muc},
          {access_persistent, muc},
          {access_admin, muc_admin},
          {max_users, 500}
         ]},
     {mod_offline,  [{access_max_user_messages,
max_user_offline_messages}]},
     {mod_privacy,  []},
     {mod_private,  []},
     {mod_proxy65,  [
          {access, local},
          {shaper, c2s_shaper}
         ]},
     {mod_pubsub,   [ % requires mod_caps
          {access_createnode, pubsub_
createnode},
          {pep_sendlast_offline, false},
          {last_item_cache, false},
          %%{plugins, [«default», «pep»]}
          {plugins, [«flat», «hometree», «pep»]}
% pep requires mod_caps
         ]},
     {mod_roster,   []},
   % Тут добавлен Общий Список Контактов (ре-
дактируется в WEB панели)
     {mod_shared_roster,[]},
     {mod_stats,     []},
     {mod_time,      []},
     {mod_vcard,     []},
     {mod_version,  []},
```

```
  % Этот модуль будет записывать все разговоры
    {mod_log_chat,  [{path, «/var/log/
ejabberd/chat»}, {format, html}]}
  ]}.
  % Конец файла ejabberd.conf
```

8) Нажимаем F2 для сохранения
9) Переходим на сайт модуля log_chat (ссылка есть в начале инструкции)
10) Выкачиваем все папки, как там, в папку mod_log_chat на свой компьютер
11) Находим — mod_log_chat/trunk/src/mod_log_chat.erl
12) Вот этот файл нам надо откомпилировать в формат – beam
13) По поводу скачивания и остального хочу сказать пару слов
Скачивать удобнее с машины, где есть графическая среда, пусть даже и Windows.
Потом просто остаётся перенести эти файлы на наш Ubuntu, а вот, как переносить - это уже дело десятое. Я буду использовать программу WinSCP (ссылка на программу есть вначале).
14) Скачали все файлы и папки с этого раздела (mod_log_chat)
15) Теперь будем переносить на Ubuntu Server
16) Запускаем WinSCP
17) Нажимаем NEW и заполняем поля IP адрес и порт
18) Что бы узнать IP нашего сервера мы должны ввести в консоле на нём команду “ifconfig” и найти например eth0 раздел inet addr:10.0.2.15
19) Порт 22 (SSH)
20) Соглашаемся на принятие ключа
21) Вводим логин – administrator со стандартным паролем
22) Получаем двухпанельный файловый менеджер
23) Кидаем с нашего компьютера в директорию /home на наш jabber сервер файлы с модулем
24) После того как сбросили все файлы отключаемся
25) Далее дело за компилированием
26) Я скинул в домашнюю директорию администратора, создав папку «1»
27) Копируем из папки «src» файл «mod_log_chat.erl» в «../trunk», тоесть на уровень выше
28) Туда же копируем необходимые для компилирования файлы из - /usr/lib/ejabberd/
29) А именно «ejabberd.hl1» и «jlib.hl1»
30) И выполняем команду, находясь в директории ./trunk - “erl -pa mod_log_chat.erl -pz ebin -make”
31) Он выдаст не критическую ошибку — Warning: behavior gen_mod undefined
32) Теперь в директории ./trunk появился файл mod_log_chat.beam
33) Копируем его в “/usr/lib/ejabberd/ebin”
34) Переходим в директорию «/var/log/ejabberd», создаём там папку «chat» и выставляем на неё полные права «sudo chmod 777 chat»
35) Выходим из «mc» и пишем «reboot»
36) После перезагрузки заводим нашего админа
37) «sudo ejabberdctl register admin jabber 123»
38) Теперь можем зайти в веб панель администратора
39) Набираем в браузере http://IP-JABBER:5280/admin
40) Где «IP-JABBER» IP нашего сервера
41) Вводим admin@jabber и пароль 123
42) В этой панели очень удобно администрировать и описывать не буду как с ней работать

НАСТРОЙКА IPTABLES

1) Итак, теперь нужно обеспечить безопасность сервера
2) Заходим на машину под пользователем administrator
3) Пишем команду sudo –s, вводим пароль
4) Открываем mc
5) Переходим в /etc/
6) Ищем файл rc.local
7) Открываем его клавишей F4
8) Дописываем перед строчкой exit 0
9) /etc/Firewall.sh
10) Нажимаем F2 и сохраняем
11) Создаём в /etc файл «Firewall.sh» нажатием клавиш Shift+F4
12) Заполняем его

```
  Firewall.sh
  #!/bin/sh
  ########## Clear all tables ##########
  iptables -t nat -F
  iptables -t filter -F
  iptables -t mangle -F

  ########## Default Rules ##########
  iptables -P INPUT DROP
  iptables -P OUTPUT ACCEPT
  iptables -P FORWARD DROP

  ########## Open Ports ##########
  # Open SSH
  iptables -t filter -A INPUT -i eth0 -p tcp –
dport 22 -j ACCEPT
  # Jabber Web Panel
  iptables -t filter -A INPUT -i eth0 -p tcp –
dport 5280 -j ACCEPT
  # Client XMPP
  iptables -t filter -A INPUT -i eth0 -p tcp –
dport 5222 -j ACCEPT

  ########## Allow by state  ##########
  iptables -t filter -A INPUT         -i eth0
-m state –state RELATED,ESTABLISHED -j ACCEPT
  iptables -t filter -A FORWARD -i eth0 -m state
–state RELATED,ESTABLISHED -j ACCEPT

  ########## Allow loopback ##########
  iptables -t filter -A INPUT         -s
localhost -d localhost -j ACCEPT
  iptables -t filter -A FORWARD -s localhost -d
localhost -j ACCEPT

  ########## Allow ICMP ##########
  iptables -t filter -A INPUT -i eth0 -p icmp
-j ACCEPT
  exit 0
```

13) Выставляем права на файл — chmod 700 /etc/Firewall.sh

КЛИЕНТ JABBER

Теперь можете установить себе клиент и начать пользоваться и настраивать свой jabber сервер

Рекомендую кроссплатформенный клиент под названием VacuumIM (ссылка в начале инструкции)

Олег Деордиев
г.Одесса

Lenovo – звук в наушниках

Установил на ноутбук Lenovo g650 Linux Ubuntu 10.04.

Настроил. Все работает – пока я не столкнулся с проблемой: при включении наушников звук продолжает идти как через наушники, так и через встроенные динамики.

Т.е. звук не отключается с динамиков при подключении наушников.

ВОЗМОЖНЫЕ ПРОБЛЕМЫ И ИХ РЕШЕНИЕ

Если звука нет, то нужно добавить в файл ***/etc/modprobe.d/alsa-base.conf*** одно из следующих значений:

Для этого в терминале:

```
sudo gedit /etc/modprobe.d/alsa-base.conf
```

Добавляем в конце строчку с одним из значений:

```
options snd-hda-intel model=»значение»
```

Для меня этого оказалось достаточно. Перепробовал я разные модели, для меня спасительным оказалась – «Ideapad»:

```
options snd-hda-intel model=»ideapad»
```

Какие еще варианты есть - можно посмотреть в оригинале статьи: http://help.ubuntu.ru/wiki/alsa

Для кого – нет – читаем дальше:

РЕШЕНИЕ :

1. Для Ubuntu 10.10 переконфигурирование ALSA ничего не даст, т.к. версия ALSA в системе уже и так последняя. Решением проблемы со звуком может послужить обновление из PPA, в терминале:

```
sudo add-apt-repository ppa:ubuntu-audio-dev/ppa
sudo apt-get update
sudo apt-get install linux-alsa-driver-modules-$(uname -r)
```

Перезагрузиться.

Если у вас в Ubuntu нет звука, не работает микрофон или неправильно настроены каналы вывода звука, то вам придётся скачать и пересобрать вручную систему вывода звука ALSA.

```
sudo apt-get install linux-headers-`uname -r` linux-backports-modules-alsa-`uname -r` linux-backports-modules-alsa-lucid-generic build-essential
sudo apt-get install gcc gawk libgettext-ruby-util libgettext-ruby1.8 libncurses5-dev libncursesw5-dev xmlto autoconf automake checkinstall
sudo apt-get upgrade
```

Установка этих пакетов займет продолжительное время и потянет за собой около 400МБ траффика. Выполнять дальнейшие инструкции без их установки бессмысленно.

2. Скачиваем исходные коды ALSA с официального сайта, желательно самые последние (на момент последнего обновления статьи 5 Сентября 2010г. это была версия 1.0.23) :

```
wget ftp://ftp.alsa-project.org/pub/driver/alsa-driver-1.0.23.tar.bz2
wget ftp://ftp.alsa-project.org/pub/lib/alsa-lib-1.0.23.tar.bz2
wget ftp://ftp.alsa-project.org/pub/utils/alsa-utils-1.0.23.tar.bz2
wget ftp://ftp.alsa-project.org/pub/firmware/alsa-firmware-1.0.23.tar.bz2
wget ftp://ftp.alsa-project.org/pub/plugins/alsa-plugins-1.0.23.tar.bz2
```

3. Распаковываем архивы:

```
tar jxvf alsa-driver-1.0.23.tar.bz2
tar jxvf alsa-lib-1.0.23.tar.bz2
tar jxvf alsa-utils-1.0.23.tar.bz2
tarjxvfalsa-firmware-1.0.23.tar.bz2
tarjxvfalsa-plugins-1.0.23.tar.bz2
```

4. Собираем и устанавливаем alsa-driver:

```
cd
rm -rf ~/.pulse*
cd ./alsa-driver-1.0.23
./configure
make
sudo make install
```

5. Собираем и устанавливаем alsa-lib:

```
cd
cd ./alsa-lib-1.0.23
./configure
make
sudo make install
```

6. Собираем и устанавливаем alsa-utils:

```
cd
sudo /sbin/alsa-utils stop
cd ./alsa-utils-1.0.23
./configure
make
sudo checkinstall
```

Внимательно смотрим, чтобы сборка alsa-utils прошла без ошибок!

7. Собираем и устанавливаем alsa-firmware:

```
cd
cd ./alsa-firmware-1.0.23
./configure
make
sudo make install
```

8. Собираем и устанавливаем alsa-plugins:

```
cd
cd ./alsa-plugins-1.0.23
./configure
make
sudo make install
```

9. Настраиваем ALSA. Выбираем нужную звуковую карту, нажимаем «ок» и на последующие вопросы отвечаем всегда «Да/Y»:

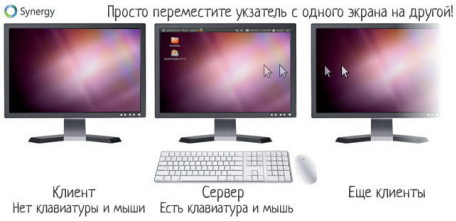
```
sudo alsaconf
```

10. Перезаружаем систему и настраиваем громкость вывода звука, ставим все на максимум (кроме PC Beep):

```
alsamixer
```

http://spider.bsyteam.net

Управление несколькими компьютерами одной клавиатурой и мышью



Представим себе ситуацию, что у вас есть ноутбук и обычный десктоп. Вот стоят они рядом включенные и вам надо работать сразу на двух. Каждый раз, когда нужно сделать что-то на ноутбуке приходится перемещать руки на его клавиатуру и тачпад? No way!

Synergy – программа, дающая возможность управлять двумя или более компьютерами, используя одну мышь и одну клавиатуру.

КАК?

Все компьютеры, которыми будем управлять, необходимо разделить на две группы: клиенты и сервер. Сервер – тот компьютер, мышь и клавиатура, которого мы будем использовать.

Для настройки можно использовать конфигурационный файл или надстройку с графическим интерфейсом. Но для начала надо установить саму программу (deb).

И да, Synergy подойдет для компьютера с Win, Lin и Mac.

1. КОНФИГ

Создаем файл synergy.conf (неважно где, при запуске можно указать местоположение файла) с содержимым (пример для двух компьютеров, компьютер2 расположен слева от компьютера1)

```
section: screens
имя_комп1:
имя_комп2:
end
section: links
имя_комп1:
left = имя_комп2
имя_комп2:
right = имя_комп1
end
```

В первой секции идет настройка всех компьютеров: двоеточие в конце строк обязательно, после него можно указывать опции. Во второй секции идет настройка положения компьютеров путем указания соседа.

Надо заметить, что имя_комп1 – именно имя компьютера, а не его IP-адрес (your-desktop вместо 192.168.0.103).

На компьютере-сервере выполняем команду:

```
synergys -f --config synergy.conf
```

Сервер заработал (или выдал сообщение об ошибке, если что-то сделано не так).

На компьютерах-клиентах выполняем:

```
synergyc -f IP_адрес
```

Вот здесь необходимо уже указать IP-адрес сервера. Пример конфиг-файла и команд запуска для двух компьютеров можно посмотреть тут (http://paste.ubuntu.com/563332/).

Топология

Пример конфигурационного-файла для Synergy.

Два компьютера:

alexander-desktop (192.168.0.103) - сервер
alexander-laptop (192.168.0.104) - клиент
Desktop - правый сосед для laptop.
Laptop - левый сосед для desktop.

Текст конфиг-файла

```
section: screens
alexander-desktop:
alexander-laptop:
end

section: links
alexander-desktop:
left = alexander-laptop
alexander-laptop:
right = alexander-desktop
end
```

Команды для запуска

На сервере (desktop):

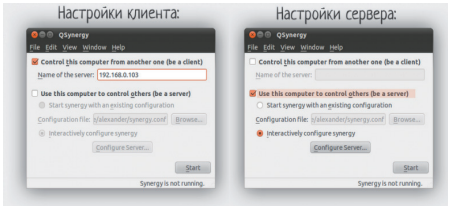
```
synergys -f --config synergy.conf
```

На клиенте (laptop):

```
synergyc -f 192.168.0.103
```

2. НАДСТРОЙКА

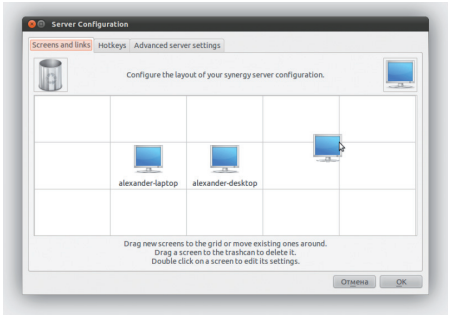
Теперь, когда вам стало понятно, как это все работает, можно перейти к более «человечной» настройке.



Загружаем и устанавливаем QSynergy – графическую утилиту для настройки (deb). Открываем, видим главное окно:

На сервере ставим вторую галочку («Use this computer to control others…»). На клиентах устанавливаем первую галочку и в поле «Name of the server» пишем IP-адрес сервера (если не знаете, то можно нажать Help About в главном окне и узнать).

Теперь о настройке сервера: для настройки можно использовать уже существующий конфиг-файл (см. пункт 1) или настроить все с нуля. Выбираем второе, нажимаем «Configure Server…».



На первой вкладке настраивается расположение и названия компьютеров. Добавить компьютер можно путем перетягивания иконки с верхнего правого угла в нужную ячейку сетки. Для переименования надо дважды нажать на иконку нужного компьютера.

Там же можно настроить «мертые углы» – углы, переход с которых на другой экран будет невозможен (не забывайте указывать размер угла). Это можно использовать, когда вы случайно переходите на другой экран из-за того, что пытаетесь закрыть окно.

На второй вкладке можно настроить клавиатурные комбинации. На третьей – дополнительные настройки сервера.

Нажимаем «ОК». Затем «Start» на компьютере-сервере и «Start» на компьютерах-клиентах.

http://unixhome.org.ua

Laptop Mode Tools
утилита для уменьшения энергопотребления ноутбука

С переходом на Linux владельцы ноутбуков могут столкнуться с проблемой высокого энергопотребления, и как следствие - малой продолжительности работы ноутбука от батареи. Однако, положение дел можно исправить. Например, можно задать таймаут отключения дисплея - 1 минуту, можно понизить яркость подсветки дисплея, что приведет к снижению энергопотребления. Но существует и еще один способ решения данного вопроса – утилита Laptop Mode Tools.

УСТАНОВКА

Пакет Laptop Mode Tools присутствует в репозиториях Ubuntu, однако, не установлен по умолчанию. Для его установки воспользуемся следующей командой:

```
sudo apt-get install laptop-mode-tools
```

Если вы используете ранние версии Ubuntu, то в ее репозиториях, скорее всего, содержится старая версия Laptop Mode Tools. Обновим ее. На 26 мая 2010 года, а также на момент написания данной статьи последняя версия – 1.55:

```
wget http://samwel.tk/laptop_mode/tools/downloads/laptop-mode-tools_1.55.tar.gz
tar xzf laptop-mode-tools_1.55.tar.gz
cd laptop-mode-tools_1.55
sudo sh ./install.sh
```

Заменим настройки старой версии:

```
sudo rm -rf /etc/laptop-mode
sudo cp -a /home/ИМЯ_ПОЛЬЗОВАТЕЛЯ/laptop-mode-tools_1.55/etc/laptop-mode /etc/
```

НАСТРОЙКА

Включаем и настраиваем Laptop Mode:

```
sudo gedit /etc/laptop-mode/laptop-mode.conf
```

Некоторые режимы уже включены по умолчанию, некоторые нам придется настроить. ***Внимательно*** ищем и заменяем значения следующих строк:

```
ENABLE_LAPTOP_MODE_ON_BATTERY=1
ENABLE_LAPTOP_MODE_ON_AC=1
ENABLE_LAPTOP_MODE_WHEN_LID_CLOSED=1
MINIMUM_BATTERY_CHARGE_PERCENT=15
DISABLE_LAPTOP_MODE_ON_CRITICAL_BATTERY_LEVEL=0
LM_BATT_MAX_LOST_WORK_SECONDS=900
LM_AC_HD_IDLE_TIMEOUT_SECONDS=1800
LM_BATT_HD_IDLE_TIMEOUT_SECONDS=600
CONTROL_HD_POWERMGMT=»1»
BATT_HD_POWERMGMT=128
LM_AC_HD_POWERMGMT=254
NOLM_AC_HD_POWERMGMT=254
```

Настроим автогибернацию при критическом заряде батареи:

```
sudo gedit /etc/laptop-mode/conf.d/auto-hibernate.conf
```

Устанавливаем параметры следующих строк:

```
ENABLE_AUTO_HIBERNATION=1
AUTO_HIBERNATION_BATTERY_CHARGE_PERCENT=15
AUTO_HIBERNATION_ON_CRITICAL_BATTERY_LEVEL=1
```

Настроим предпочтительные частоты процессора:

```
sudo gedit /etc/laptop-mode/conf.d/cpufreq.conf
```

Устанавливаем параметры следующих строк:

```
CONTROL_CPU_FREQUENCY=»1»
LM_AC_CPU_IGNORE_NICE_LOAD=0
```

Настроим Gnome Power Manager, в терминале либо через Alt+F2 вводим:

```
gconf-editor
```

Переходим в раздел ***apps – gnome-power-manager – thresholds***, устанавливаем следующие значения:

```
percentage_action = 15
percentage_critical = 15
percentage_low = 30
time_action = 900
time_critical = 900
time_low = 1500
```

В разделе ***apps – gnome-power-manager – actions*** устанавливаем следующие значения:

```
critical_battery = hibernate
sleep_type_ac = suspend
sleep_type_battery = hibernate
```

На этом настройка закончена. Перезагружаемся и проверяем, работает ли Laptop Mode Tools:

```
cat /proc/sys/vm/laptop_mode
```

Если выводится значение отличное от нуля, значит Laptop Mode установлен правильно.

Сергей Луконин.
http://linux-easy.ru

ITOutsourcing

Доверьте свои сервера профессионалам

http://itoutsourcing.org.ua

+38 (050) 022-33-61

Vim: необходимый минимум знаний

Пользуетесь ***gEdit***? Прекрасный редактор. Но, что вы будете делать, когда понадобится изменить пару конфигурационных файлов не сервере, расположенном в соседней области? Когда речь заходит о системном администрировании, все дороги ведут обратно к Vim! Никогда им не пользовались или, ещё хуже, вообще пугаетесь, когда слышите его имя? На самом деле, не так уж он страшен и непонятен, как полагают некоторые. И потом, чтобы решать задачи, вроде редактирования файлов конфигурации, совсем не нужно изучать всю бесконечную тучу возможностей этого, поистине мощного редактора. Вам понадобится знать совсем немного команд, чтобы быстро и без проблем сделать свою работу. Конечно, человек, хорошо знающий Vim, сделает это ещё быстрее и эффективнее, но ведь все же с чего-то начинали, правда?

Вообще, когда вы работаете на сервере с Vim (который по сути является клоном редактораVi), скорее всего, вы будете иметь дело в «vim-tiny» – урезанной версией Vim, в которой оставлен только набор функций «классического» редактора Vi.

РЕЖИМ ВСТАВКИ

Первая вещь, на которую вы должны обратить внимание – это то, что Vim может работать в различных режимах: командном режиме, режиме вставки, режиме «последней строки» (известном ещё как «ex mode»). Непосредственно после запуска, Vim находится в командном режиме. В этом режиме весь ввод с клавиатуры будет воспринят как команды редактору, а не редактирование текста. Чтобы переключиться в режим вставки, нажмите 'i', после чего вы получите возможность редактировать текст. При помощи нажатия на Esc вы можете затем вернуться в командный режим, а введя ':' в командном режиме, вы попадаете вех-режим.

КОМАНДНЫЙ РЕЖИМ

Команды, которые могут использоваться в командном режиме, можно перечислять очень долго. Но пока что мы ограничимся лишь самым необходимым. Давайте начнём с перемещения по тексту:

'h' – перемещение на один символ влево;
'l' – перемещение на один символ вправо;
'j' – перемещение на один символ вниз;
'k' – перемещение на один символ вверх;

Можно смещать курсор не на один символ, а на слово целиком: ***'w'*** перемещает на слово вперёд, а ***'b***' – на слово назад. Словом в ***Vim*** считается любая последовательность алфавитно-цифровых символов. Например, ***«word»*** – это слово, а ***«one-year»*** – это уже два слова, поскольку они разделены дефисом.

Удаление выполняется при помощи команд ***'d***' или ***'x***'. Если вам нужно удалить один символ, поставьте курсор над этим символом и нажмите ***'x'***. Для удаления слова под курсором используйте '***dw***', а для удаления предыдущего слова – '***db***'.

Если требуется удалить целую строку, используйте команду '***dd'.*** При помощи команды ***'d$***' можно удалить часть строки от позиции курсора до конца строки, а командой ***'d^***' (или ***'d0***') – от текущей позиции курсора до начала строки.

КОПИРОВАНИЕ И ВСТАВКА

Копирование и вставку текста в ***Vim*** можно выполнять невероятно быстро. Чтобы переключиться в режим выделения текста, используйте команду ***'v', 'V'*** или ***'Ctrl+v'***. После переключения в режим выделения вы можете выделять текст, перемещая курсор клавишами ***h, l, j, k*** или же клавишами со стрелками. В режиме ***'v'*** – будет происходить «обычное» выделение, в режиме ***'V'*** – выделение целых строк, а в режиме '***Ctrl+v'*** – блочное выделение, с его помощью можно выделять вертикальные блоки.

После того, как вы выделите нужный текст, используйте команду ***'y'*** для того, чтобы скопировать выделенный фрагмент в буфер.

Далее, при помощи команды ***'p'*** или ***'P'*** можно вставить скопированный текст из буфера. ***'p***' вставит текст после курсора, а ***'P'*** – перед ним.

ПОИСК И ЗАМЕНА

Чтобы переключиться в режим поиска используйте команду ***'/'*** – для поиска в прямом направлении, или команду ***'?***' – в обратном; затем введите шаблон поиска.

Поиск и замена в тексте выполняется при помощи команды ***':s***' с заданной областью поиска и замены. Например, чтобы найти ***«old»*** и заменить на ***«new»***, используйте следующую команду:

```
:%s/old/new/
```

Символ процента перед ***'s'*** означает глобальную область поиска. Вместо глобальной области вы можете определить диапазон строк, например:

```
:1,15s/old/new/
```

Имейте ввиду, что указанные выше команды, заменяют только первое найденное вхождение. Часто бывает нужно найти и заменить все найденные вхождения в тексте. Для этого можно воспользоваться «глобальной» заменой, добавив к команде букву ***'g'***:

```
:%s/old/new/g
```

Приведённая выше команда заменит все найденные ***«old»*** на ***«new»***. Дополнительно вы можете заставить ***Vim*** запрашивать у вас подтверждение перед выполнением замены:

```
:%s/old/new/gc
```

ОТКАТ ДЕЙСТВИЙ

Что делать, если вы внесли изменения, которые не нужно было вносить? Воспользуйтесь командой ***'u'***, которая отменит последнее изменение, сделанное вами. Если же после отката на действие назад вы снова передумали и решили «отменить отмену», вам поможет команда ***'Ctrl+r'***, которая делает откат в обратную сторону.

Имейте ввиду, что если ваш ***Vim*** работает в режиме совместимости с ***Vi***, то вы не можете откатиться более, чем на один шаг назад. В своём же «родном» режиме ***Vim*** «помнит» и позволяет откатываться на большое количество шагов.

СОХРАНЕНИЕ И ВЫХОД

Один из вопросов, который возникает у новичков, первый раз работающих с ***Vim***, звучит примерно так: «как мне выйти из него?». Действительно, глядя на экран с редактором, догадаться, как именно это делается, невозможно. Это нужно просто знать.

Если вы хотите сохранить внесённые в файл изменения, используйте команду ***':w'***, а чтобы сохранить изменения и покинуть редактор – команду ***':wq'***. Не хотите сохранять внесённые исправления? Бывает и такое. При помощи команды ***':q!'*** можно выйти из ***Vim***, при этом, не сохраняя текущий файл.

Также, если нужно, вы можете сохранить редактируемый текст в другой файл при помощи команды ***':w имя_файла'***.

ЧТО ДАЛЬШЕ

Такое вот кратенькое руководство получилось. Надеюсь, оно поможет многим начать работу с Vim и приобщиться к этому мощному инструменту, зачастую незаменимому в некоторых ситуациях. Если у вас возникнет желание дальше изучать возможности, предлагаемые этим редактором, обязательно почитайте руководство ***Vim***, которое можно запустить из командной строки вызовом ***vimtutor***. Ко всему прочему, документацию по ***Vim*** можно почитать из самого редактора, воспользовавшись командой ***':help'.***

По мотивам serverwatch.com
Александр Шепетко
http://www.ashep.org

«А знаете ли вы, что ...?» или несколько простых приемов для работы с консолью

Не все любят работать с консолью, предпочитая этому GUI-интерфейсы. Для многих же — консоль является неотъемлемой частью ежедневной работы в Linux. Зачастую новички просто не знают о методах, позволяющих сделать работу в консоли максимально комфортной, а иногда даже более удобной, нежели через графический интерфейс. Давайте рассмотрим несколько простых, но эффективных приемов для работы с консолью в Linux, в форме вопросов «А знаете ли Вы, что ...?».

А ЗНАЕТЕ ЛИ ВЫ, ЧТО ...?

...если вы выполнили команду и забыли про 'sudo', то можете просто набрать '***sudo !!***' для вторичного запуска предыдущей команды с использованием 'sudo'.

...для перемещения курсора в начало строки можно использовать сочетание ***Ctrl+A***, а для перемещения в конец — сочетание ***Ctrl+E***.

...вместо команды 'exit' для закрытия терминала или сессии вы можете использовать комбинацию ***Ctrl+D***.

...для автоматического дополнения вводимой команды вы можете использовать клавишу ***Tab***. А для вывода возможных вариантов — дважды нажать ***Tab***.

...с помощью команды ***free*** можно быстро узнать, сколько ОЗУ и swap занято, и сколько свободно. А по команде top можно получить информацию о запущенных в системе процессах. «Убить» зависший процесс можно с помощью команды ***killall имя_процесса***.

...если вы ошиблись при наборе команды, то можете удалить последнее слово с помощью сочетания ***Ctrl+W***, или же всю строку с помощью ***Ctrl+U***.

...для повторения произвольной команды через постоянные интервалы времени с выводом временных отметок вы можете воспользоваться командой ***watch*** (подробнее читайте в ***man watch***).

...для поиска ранее введенных команд вы можете использовать сочетание ***Ctrl+R***, после чего ввести известные символы из искомой команды.

...с помощью команды ***history*** вы можете посмотреть список последних набранных вами команд.

...если вы ошиблись в одной или нескольких символах при написании команды, или вам просто нужно повторить команду с использованием других параметров, воспользуйтесь символом '^'. Например, вы набрали команду *gistory* вместо *history*, тогда просто наберите ***^g^h***, чтобы заменить символ '***g***' на '***h***' и повторно выполнить команду.

Сергей Луконин
http://linux-easy.ru

Основы линукс. Часть 1

О ЧЕМ СТАТЬЯ

Прочитав эту статью, вы узнаете что такое bash (стандартный командный интерпретатор линукс), научитесь обращаться со стандартными командами: ***ls, cp, mv...*** поймете назначение инодов, жестких и символических ссылок и многое другое.

Это пособие предназначено для новичков в линукс и для тех, кто хочет повторить или усовершенствовать понимание основных принципов линукс, таких как копирование и перемещение файлов, создание ссылок, использование стандартных команд линукс наряду с перенаправлениями и пайпами. В этой статье вы найдете множество примеров, поясняющих изложенный материал. Для начинающих большинство информации окажется новой, а для более продвинутых пользователей этот материал может стать прекрасным пособием для обобщения имеющихся знаний и навыков.

ВВЕДЕНИЕ В BASH

Shell

Если вы используете линукс, то знаете что после логина вас приветствует приглашение командного интерпретатора. Например такое:

```
$
```

Если после логина загружается графическая оболочка, то чтобы добраться до командного интерпретатора нужно запустить эмулятор терминала (***gnome-terminal, xfce4-terminal, konsole, xterm, rxvt...***) или переключиться на один из виртуальных терминалов нажав***Ctrl–Alt–F1*** или ***Ctrl–Alt–F2*** и т.д.

Приглашение командного интерпретатора на вашем компьютере может отличаться от того что показано в примере. Оно может содержать имя пользователя, имя компьютера и название текущей рабочей директории. Но, несмотря на все эти различия, программа, которая печатает это приглашение называется «shell» (оболочка), и скорее всего в роли вашей командной оболочки выступает программа, которая называется bash.

У ВАС ЗАПУЩЕН BASH?

Проверить, запущен ли bash, можно следующей командой:

```
$echo $SHELL
/bin/bash
```

Если в результате выполнения этой команды вы получили ошибку или её вывод отличается от того, что в примере, то возможно в вашей системе в качестве командной оболочки используется не bash. Несмотря на это, большая часть материала будет актуальна, но все же рекомендуем вам переключиться на bash. Сделать это можно (если bash установлен в системе) командой:

```
$bash
```

Что такое bash

Bash (акроним от «Bourne-again SHell») это стандартный интерпретатор команд на большинстве линукс систем. В его обязанности входит обработка и исполнение команд, с помощью которых пользователь управляет компьютером. После того, как вы завершили работу, можно завершить процесс командного интерпретатора. После нажатия клавиш ***Ctrl–D***, команд exit или logout процесс командного интерпретатора будет завершен и на экране снова появится приглашение ввести имя пользователя и пароль.

ИСПОЛЬЗОВАНИЕ «CD»

Давайте начнем использовать bash для навигации по файловой системе. Для начала напечатайте следующую команду:

```
$cd /
```

Этой командой мы указали bash-у что хотим переместиться в корневую директорию — /. Все директории в системе организованы в древовидную структуру и / это её начало (или корень). Команда cd служит для изменения текущей рабочей директории.

ПУТИ

Чтобы узнать в каком месте файловой системы в данный момент вы находитесь (текущую рабочую директорию) наберите:

```
$pwd /
```

В приведенном выше примере / – аргумент команды ***cd*** – называется путь. Это место файловой системы, куда мы хотим переместиться. В данном случае / – абсолютный путь. Это значит, что путь указан относительно корневой директории.

Абсолютные пути

Вот несколько примеров абсолютных путей

```
/dev
/usr
/usr/bin
/usr/local/bin
```

Как вы уже могли заметить, все эти пути объединяет то, что они начинаются ***с /.*** Указывая путь ***/usr/local/bin*** в качестве аргумента команде cd , мы говорим ей перейти в корневую директорию /, затем в директорию usr, потом в ***loca***l и ***bin.*** Абсолютные пути всегда начинаются ***с /***

Относительные пути

Второй вид путей называется относительными. Bash, команда ***cd*** и другие команды отсчитывают эти пути относительно текущей директории. Относительные пути никогда не начинаются ***с /***. Например, если мы находимся в ***/usr***

```
$ cd /usr
```

Затем мы можем перейти в ***/usr/local/bin***, используя относительный путь

```
$cd local/bin
$pwd
/usr/local/bin
```

ИСПОЛЬЗОВАНИЕ «..»

Относительные пути могут содержать одну или несколько директорий ***«..».*** ***«..»*** указывает на родительскую директорию по отношению к нашей рабочей директории. Пример:

```
$pwd
/usr/local/bin
$cd ..
```

```
$pwd /usr/local
```

Как вы видите, команда ***cd ..*** 'поднимает нас на уровень выше'.

Можно добавить .. к относительному пути. Это позволит переместиться в директорию, которая находится на одном уровне с той, в которой мы находимся. Пример:

```
$pwd
/usr/local
$cd ../share
$pwd
/usr/share
```

ПРИМЕРЫ С ИСПОЛЬЗОВАНИЕМ ОТНОСИТЕЛЬНЫХ ПУТЕЙ

Относительные пути могут быть довольно сложными. Вот несколько примеров. Результат выполнения команд не показан, попробуйте определить в какой директории вы окажетесь, используя bash.

```
$cd /bin
$cd ../usr/share/zoneinfo
$cd /usr/X11R6/bin
$cd ../lib/X11
$cd /usr/bin
$cd ../bin/../bin
```

РАБОЧАЯ ДИРЕКТОРИЯ «.»

Перед тем, как закончить разговор о команде ***cd***, следует упомянуть еще несколько вещей. Во-первых, существует ещё одна специальная директория ***«.»***, которая указывает на текущую директорию. Эта директория используется для запуска исполняемых файлов, находящихся в текущей директории.

$./MYPROG

В последнем примере myprog - это исполняемый файл, находящийся в текущей директории, который будет запущен на исполнение.

cd и домашняя директория пользователя

Для того, чтобы перейти в домашнюю директорию, нужно набрать

```
$cd
```

Без аргумента ***cd*** переместит вас в домашнюю директорию. Для суперпользователя домашней обычно является директория ***/root,*** а для обычных пользователей — ***/home/username***/. Но что, если мы хотим указать конкретный файл, находящийся в домашней директории. Например, как аргумент к программе ***'myprog'***? Можно написать:

```
$./myprog /home/user/myfile.txt
```

Однако, использовать абсолютные пути к файлам не всегда удобно. Эту же операцию можно сделать при помощи ~ –тильды:

```
$./myprog ~/myfile.txt
```

~ — специальное имя, указывающее в bash на домашнюю директорию пользователя.

ДОМАШНИЕ ДИРЕКТОРИИ ДРУГИХ ПОЛЬЗОВАТЕЛЕЙ

Но что, если нам нужно указать файл в домашней директории другого пользователя? Для этого после тильды нужно указать имя этого пользователя. Например, чтобы указать на файл fredsfile.txt, находящийся в домашней директории пользователя fred:

```
$./myprog ~fred/fredsfile.txt
```

КОМАНДЫ ЛИНУКС ВВЕДЕНИЕ В LS

Вероятно, вы уже знакомы с командой ls, которая, вызванная без аргументов, выводит на экран список файлов, хранящихся в рабочей директории:

```
$cd /usr
$ls
X11R6           doc
i686-pc-linux-gnu       lib
      man
sbin            ssl
bin
gentoo-x86  include
libexec         portage
share                   tmp
      distfiles
i686-linux      info
local
portage.old
src
```

Если указать опцию -a, можно будет увидеть все файлы, включая скрытые (имена которых начинаются с точки).

```
$ls -a
.         bin
gentoo-x86  include
libexec         portage
share tmp
  ..        distfiles
i686-linux  info
local       portage.old  src
X11R6
  doc i686-pc-linux-gnu  lib
man         sbin         ssl
```

ПОДРОБНЫЙ СПИСОК ДИРЕКТОРИЙ

После самой команды ls в качестве ее аргумента можно указать один или более файлов или директорий. Если указать имя файла, то команда ls выведет информацию только об этом файле. А если указать название директории, ls покажет все ее содержимое. Опция '-l' команды ls бывает очень полезной, если вы хотите, кроме имен файлов узнать более подробную информацию о них (права на файл, имя владельца, время последнего изменения файла и его размер).

В следующем примере показано применение опции '-l' для вывода информации о файлах, хранящихся в директории /usr

```
$ls -l /usr
drwxr-xr-x       7     root
root  168 Nov 24 14:02 X11R6
drwxr-xr-x       2     root
root  14576 Dec 27 08:56 bin
drwxr-xr-x       2     root
root  8856  Dec  26  12:47
distfiles
lrwxrwxrwx       1     root
root  9 Dec 22 20:57 doc ->
share/doc
drwxr-xr-x       62    root
root  1856  Dec  27  15:54
gentoo-x86
drwxr-xr-x       4     root
root  152 Dec 12 23:10 i686-
linux
drwxr-xr-x       4     root
root  96 Nov 24 13:17 i686-
pc-linux-gnu
drwxr-xr-x       54    root
root  5992  Dec  24  22:30
include
lrwxrwxrwx       1     root
root  10 Dec 22 20:57 info ->
share/info
drwxr-xr-x       28    root
root  13552 Dec 26 00:31 lib
drwxr-xr-x       3     root
```

```
root   72 Nov 25 00:34 libexec
   drwxr-xr-x       8      root
root   240 Dec 22 20:57 local
   lrwxrwxrwx       1      root
root   9 Dec 22 20:57 man ->
share/man
   lrwxrwxrwx       1      root
root   11 Dec 8 07:59 portage
-> gentoo-x86/
   drwxr-xr-x       60     root
root   1864    Dec    8    07:55
portage.old
   drwxr-xr-x       3      root
root   3096 Dec 22 20:57 sbin
   drwxr-xr-x       46     root
root   1144 Dec 24 15:32 share
   drwxr-xr-x       8      root
root   328 Dec 26 00:07 src
   drwxr-xr-x       6      root
root   176 Nov 24 14:25 ssl
   lrwxrwxrwx       1      root
root   10 Dec 22 20:57 tmp ->
../var/tmp
```

В первой колонке показана информация о правах доступа к каждому файлу в списке. (Немного позже я объясню, какая буква что обозначает) Следующая колонка показывает количество ссылок на каждый элемент списка. Третья и четвертая колонки — владелец и группа файла, соответственно. Пятая колонка — размер. Шестая — время последнего изменения файла ('last modified time' или mtime). Последняя колонка — имя файла или директории (Если это ссылка, то после знака '->' стоит имя объекта, на который она ссылается).

КАК ПОСМОТРЕТЬ ТОЛЬКО ДИРЕКТОРИИ

Иногда возникает потребность посмотреть информацию только о директориях, а не о всем их содержимом. С этой задачей поможет справиться опция '-d', которая указывает команде выводить информацию только о директориях. Пример:

```
   $ls -dl /usr /usr/bin /usr/
X11R6/bin ../share
   drwxr-xr-x       4      root
root   96   Dec   18   18:17   ../
share
   drwxr-xr-x       17     root
root   576 Dec 24 09:03 /usr
   drwxr-xr-x       2      root
root   3192 Dec 26 12:52 /usr/
X11R6/bin
   drwxr-xr-x       2      root
root   14576  Dec  27  08:56  /
usr/bin
```

РЕКУРСИВНЫЙ СПИСОК И ИНФОРМАЦИЯ О ИНОДАХ

Действие опции ***'-R'*** противоположно действию ***'-d'***. Она позволяет выводить информацию о файлах, находящихся в директории рекурсивно. Сначала показывается содержимое директории верхнего уровня, потом по очереди содержимое всех поддиректорий и так далее. Вывод этой команды может быть достаточно объемным, поэтому мы не приводим ее пример, но вы можете попробовать сделать это самостоятельно, набрав в командной строке ***'ls -R'*** или ***'ls -Rl'***.

И, наконец, опция '-i' используется для вывода инодов каждого объекта файловой системы.

```
   $ls -i /usr
   1409 X11R6
314258 i686-linux  4 3 0 9 0
libexec      13394 sbin
1417 bin
   1513 i686-pc-linux-gnu
5120 local         1 3 4 0 8
share 8316 distfiles
1517 include
   776 man         23779 src
      43 doc
1386 info          9 3 8 9 2
portage
   36737 ssl
70744 gentoo-x86   1585 lib
5132 portage.old   784 tmp
```

ЧТО ТАКОЕ ИНОДЫ?

Каждый объект файловой системы (файл, директория...) имеет свой уникальный номер, называемый инодом (inode number). Эта информация может показаться незначительной, но понимание функции инодов поможет вам разобраться во многих операциях над файловой системой. Например, посмотрим на **«.»** и **«..»** как на ссылки, присутствующие в каждой директории. Чтобы понять, что из себя представляет директория «..»,, узнаем инод директории ***/use/local***

```
   $ls -id /usr/local
   5120 /usr/local
```

Как можем видеть, инод директории ***/usr/local — 5120***. Теперь посмотрим какой инод у директории ***/usr/local/bin/..:***

```
   $ls -id /usr/local/bin/..
   5120 /usr/local/bin/..
```

Получается, что иноды директорий ***/usr/local u /usr/local/bin/.***. совпадают! Это значит, что на инод ***5120*** ссылаются два имени: ***/usr/local u /usr/local/bin/..*** Тоесть - это два разных имени одной директории. Каждый инод указывает на определенное место на диске.

С каждым инодом может быть связано несколько имен объектов файловой системы. Количество 'синонимов' файла (объектов файловой системы, ссылающихся на один инод) показывает число во втором столбце вывода команды 'ls -l'.

```
   $ ls -dl /usr/local
   drwxr-xr-x       8      root
root   240 Dec 22 20:57 /usr/
local
```

На этом примере видно (второй столбец), что на директорию /usr/local ссылаются 8 разных объектов файловой системы. Вот их имена:

```
   /usr/local
   /usr/local/.
   /usr/local/bin/..
   /usr/local/games/..
   /usr/local/lib/..
   /usr/local/sbin/..
   /usr/local/share/..
   /usr/local/src/..
```

MKDIR

Давайте рассмотрим команду mkdir. Она служит для создания новых директорий. В следующем примере демонстрируется создание трех новых директорий (tic, tac, toe) в директории /tmp

```
   $cd /tmp
   $mkdir tic tac toe
```

По умолчанию команда mkdir не может создать вложенной структуры директорий. Поэтому, если вам нужно создать несколько вложенных одна в другую директорий (won/der/ful), то вам придется три раза поочередно вызывать эту команду:

```
   $mkdir won/der/ful
```

```
mkdir: cannot create directory 'won/der/ful': No such file or directory
$mkdir won
$mkdir won/der
$mkdir won/der/ful
```

Упростить эту операцию можно добавив опцию '-p' к команде mkdir. Эта опция позволяет создавать вложенную структуру директорий:

```
$mkdir -p easy/as/pie
```

Чтобы узнать о возможностях этой утилиты подробнее, прочитайте справку, которая вызывается командой man mkdir. Справки есть практически ко всем командам из этого руководства (например man ls), кроме cd, т.к. она встроена в bash (для таких команд справка вызывается так: help cd)

TOUCH

Перейдем к изучению команд cp и mv, служащих для копирования, переименования и перемещения файлов и директорий. Но перед этим создадим пустой файл в директории /tmp при помощи команды touch:

```
$cd /tmp
$touch copyme
```

Команда touch обновляет время последнего доступа к файлу (шестая колонка вывода команды ls -l) если он уже существует или создает новый пустой файл, если его ещё нет. После этой операции у нас должен появиться пустой файл /tmp/copyme.

ECHO

Теперь, когда у нас есть пустой файл, запишем в него текстовую строку при помощи команды echo, которая выводит переданный ей аргумент на стандартное устройство вывода (текстовый терминал в нашем случае).

```
$echo «firstfile»
firstfile
```

Чтобы записать строку в наш файл, перенаправим в него вывод команды ***echo:***

```
$echo «firstfile» > copyme
```

Знак > (больше) указывает командной оболочке, что нужно перенаправить вывод команды стоящей слева в файл, имя которого находится справа. Если файла с таким именем не существует, он будет создан автоматически. А если такой файл уже есть, то он будет перезаписан (все его содержимое будет стерто перед записью нашей строки). Команда ***'ls -l'***покажет, что размер нашего файла теперь равен 10 байтам — девять байт занимает слово 'firstfile' и один байт символ перевода строки.

```
$ ls -l copyme
-rw-r--r--              1 root root    10 Dec 28 14:13 copyme
```

CAT И CP

Для вывода содержимого файла на терминал используется команда cat:

```
$cat copyme
firstfile
```

Теперь мы можем приступить к разбору базовой функциональности команды cp. Эта команда принимает два аргумента. Первый — имя уже существующего файла ('copyme'), второй — название новой копии, которую мы хотим сделать ('copiedme').

```
$cp copyme copiedme
```

Можем убедиться, что новая копия файла имеет другой номер инода (это значит, что мы получили действительно новый отдельный файл, а не просто ссылку на старый)

```
$ls -i copyme copiedme
648284 copiedme 6 5 0 7 0 4 copyme
```

MV

Теперь применим команду mv, чтобы переименовать файл (***«copiedme» –> «movedme»***). Номер инода после этой операции не меняется, а изменяется только название файла.

```
$mv copiedme movedme
$ls -i movedme
648284 movedme
```

Номер инода не изменяется только при условии, что переименованный файл остается в пределах той файловой системы где находился исходный файл. Мы рассмотрим подробнее устройство файловых систем в одной из следующих частей этого пособия.

Команда mv позволяет не только переименовывать файлы, но и перемещать их. Например, чтобы переместить файл ***/var/tmp/myfile.txt*** в директорию ***/home/user*** нужно дать команду:

```
$mv /var/tmp/myfile.txt /home/user
```

Файл будет перемещен в домашнюю директорию пользователя user даже, если она находится в другой файловой системе (в этом случае файл будет скопирован в новое место, после чего оригинал будет удален). Как вы могли уже догадаться, перемещение файла в другую файловую систему приводит к изменению его инода. Это происходит потому, что каждая файловая система имеет свой отдельный набор инодов.

Нужно заметить, существует вероятность, что новый присвоенный номер инода может совпасть со старым, но она чрезвычайно мала.

Чтобы переместить одновременно несколько файлов в одну директорию нужно написать:

```
$mv /var/tmp/myfile1.txt /var/tmp/myfile2.txt /home/user
```

или

```
$mv -t /home/user /var/tmp/myfile1.txt /var/tmp/myfile2.txt
```

Если добавить опцию '-v', на экран будет выведен отчет о проделанной операции:

```
$ mv -vt /home/user /var/tmp/myfile1.txt /var/tmp/myfile2.txt
'/var/tmp/myfile1.txt' -> '/home/user/myfile1.txt'
'/var/tmp/myfile2.txt' -> '/home/user/myfile2.txt'
```

СОЗДАНИЕ ССЫЛОК И УДАЛЕНИЕ ФАЙЛОВ ЖЕСТКИЕ ССЫЛКИ

Я уже упоминал слово «ссылка», когда говорил о директориях и инодах. На самом деле в линуксе существует два вида ссылок. Первый вид называют жесткими ссылками. Каждый инод может иметь несколько, связанных с ним жестких ссылок. Таким образом, получается, что файл присутствует в системе под несколькими

разными именами. Файл существует до тех пор, пока с его инодом связано хотя бы одно имя. Понятия «жёсткая ссылка на файл» и «имя файла» являются синонимами. Новые жесткие ссылки на файл можно сделать при помощи команды ln

```
$cd /tmp
$touch firstlink
$ln firstlink secondlink
$ls -i firstlink secondlink
15782 firstlink 1 5 7 8 2 secondlink
```

Как видно из примера, жесткие ссылки работают на уровне инодов, указывая на определенный файл. В линуксе у жестких ссылок есть несколько ограничений. Во-первых, вы можете создавать жесткие ссылки только на файлы, но не на директории. Вот именно, несмотря на то, что в системе существуют жесткие ссылки на директории ('.' и '..'), даже суперпользователь не может создавать дополнительные жесткие ссылки на директории. Во-вторых, невозможно создать жесткую ссылку на файл, находящийся в другой файловой системе, т.к. каждая файловая система имеет свой уникальный набор инодов.

СИМВОЛИЧЕСКИЕ ССЫЛКИ

На практика чаще применяют символические ссылки (или симлинки). Симлинк - это специальный вид файла, который ссылается на другой файл по имени, а не напрямую на инод. Симлинки не предохраняют файл от удаления. Если файл удалить, то симлинк на него станет нерабочим (или битым).

Симлинки создаются командой ln с опцией '-s':

```
$ln -s secondlink thirdlink
$ls -l firstlink secondlink thirdlink
-rw-rw-r--  2 agriffis agriffis  0 Dec 31 19:08 firstlink
-rw-rw-r--  2 agriffis agriffis  0 Dec 31 19:08 secondlink
lrwxrwxrwx  1 agriffis agriffis 10 Dec 31 19:39 thirdlink -> secondlink
```

Символическую ссылку можно распознать по выводу команды ls -l: во-первых, в первой колонке у симлинков стоит буква 'l' (первая буква английского слова link–ссылка), во-вторых, размер симлинка равен количеству букв в имени файла, на который он ссылается ('secondlink' в нашем случае), в-третьих, последняя колонка помимо имени ссылки содержит имя файла , на который она ссылается после знака –>

ПОДРОБНЕЕ О СИМЛИНКАХ

Символические ссылки намного гибче жестких. С их помощью вы можете ссылаться на любой объект (файл, директория, сокет...) любой файловой системы.

Рассмотрим ситуацию, когда мы хотим сделать симлинк, который указывает на /usr/local/bin и находится в директории /tmp/. Мы можем написать:

```
$ln -s /usr/local/bin bin1
$ls -l bin1
lrwxrwxrwx  1 root root  14 Jan 1 15:42 bin1 -> /usr/local/bin
```

Или

```
$ln -s ../usr/local/bin bin2
$ls -l bin2
lrwxrwxrwx  1 root root  16 Jan 1 15:43 bin2 -> ../usr/local/bin
```

Как видно из этих примеров, обе ссылки указывают на одну директорию. Но, если вторую ссылку переместить из /tmp в другую директорию, она может оказаться битой из-за использованного в ней относительного пути.

```
$ls -l bin2
lrwxrwxrwx  1 root root  16 Jan 1 15:43 bin2 -> ../usr/local/bin
$mkdir mynewdir
$mv bin2 mynewdir
$cd mynewdir
$cd bin2
bash: cd: bin2: No such file or directory
```

Так как не существует директории /tmp/usr/local/bin/, мы не сможем сменить рабочую директорию на bin2; другими словами, после перемещения ссылка перестала работать (стала 'битой').

По этой причине, иногда стоит избегать создания симлинков, используя относительные пути. Но иногда это бывает удобно. Рассмотрим такой случай: допустим, что мы хотим сделать ссылку на программу в /usr/bin (или другими словами присвоить этой программе альтернативное имя):

```
#ls -l /usr/bin/keychain
-rwxr-xr-x  1 root root  10150 Dec 12 20:09 /usr/bin/keychain
```

Суперпользователь (root) может захотеть сделать ссылку на программу «keychain» с более коротким именем «kc». В этом примере у нас есть рутовый доступ к системе, о чем свидетельствует приглашение bash, изменившееся на «#». Нам нужны права суперпользователя потому, что обычные пользователи не могут создавать файлы в директории /usr/bin/. Теперь мы можем от имени рута создать альтернативное имя для нашей программы:

```
#cd /usr/bin
#ln -s /usr/bin/keychain kc
#ls -l keychain
-rwxr-xr-x  1 root root  10150 Dec 12 20:09 /usr/bin/keychain
#ls -l kc
lrwxrwxrwx  1 root root  17 Mar 27 17:44 kc -> /usr/bin/keychain
```

В этом примере мы создали симлинк kc, ссылающийся на файл /usr/bin/keychain.

Эта ссылка полностью рабочая, но она перестанет работать, если мы решим перенести оба файла 'keychain' и 'kc' из директории /usr/bin/ в /usr/local/bin/:

```
#mv /usr/bin/keychain /usr/bin/kc /usr/local/bin
#ls -l /usr/local/bin/keychain
-rwxr-xr-x  1 root root  10150 Dec 12 20:09 /usr/local/bin/keychain
#ls -l /usr/local/bin/kc
```

```
lrwxrwxrwx 1 root root 17 Mar 27 17:44 kc -> /usr/bin/keychain
```

Из-за того что мы использовали абсолютный путь при создании ссылки, она продолжает указывать на файл /usr/bin/keychain, которого больше нет. Но если бы мы использовали относительный путь при создании ссылки, она бы осталась рабочей.

Можно сделать вывод, что ссылки, созданные с абсолютными и относительными путями имеют каждая свое применение. Поэтому при создании симлинка нужно выбрать способ, который будет уместнее в данной конкретной ситуации.

Часто оба вида симлинков (с абсолютными и относительными путями) работают нормально. Следующий пример показывает способ создания симлинка, который продолжает работать после перемещения его и файла, на который он ссылается в другую директорию:

```
#cd /usr/bin
#ln -s keychain kc
#ls -l kc
lrwxrwxrwx 1 root root 8 Jan 5 12:40 kc -> keychain
#mv keychain kc /usr/local/bin
#ls -l /usr/local/bin/keychain
-rwxr-xr-x 1 root root 10150 Dec 12 20:09 /usr/local/bin/keychain
#ls -l /usr/local/bin/kc
lrwxrwxrwx 1 root root 17 Mar 27 17:44 kc -> keychain
```

Теперь мы можем запускать программу 'keychain', обратившись к ней по имени ***/usr/local/bin/kc***

RM

Теперь, когда мы знаем, как работают команды cp, mv и ln пришло время узнать, как удалять файлы. Обычно, удаление производится при помощи команды rm. Чтобы удалить несколько файлов, просто укажите их имена через пробел в командной строке как аргументы rm:

```
$cd /tmp
$touch file1 file2
$ls -l file1 file2
-rw-r--r-- 1 root root 0 Jan 1 16:41 file1
-rw-r--r-- 1 root root 0 Jan 1 16:41 file2
$rm file1 file2
$ls -l file1 file2
ls: file1: No such file or directory
ls: file2: No such file or directory
```

Помните, что удаленные файлы невозможно восстановить (хотя можно и попробовать). Поэтому многие начинающие пользователи линукс используют опцию '-i' команды rm, которая требует запрашивать у пользователя подтверждение удаления каждого файла.

```
$rm -i file1 file2
rm: remove regular empty file `file1'? y
rm: remove regular empty file `file2'? y
```

В последнем примере перед удалением каждого файла команда rm спрашивает: действительно ли пользователь хочет удалить файл? Чтобы подтвердить удаление, нужно нажать клавишу «y» на клавиатуре, а чтобы отказаться от удаления — клавишу «n».

Прервать выполнение любой команды (если что-то пошло не так, как задумывалось) можно, нажав комбинацию ***Ctrl–C.***

Сделать так, чтобы команда rm запрашивала подтверждение на удаление каждого файла даже без опции '-i' можно добавив в файл ~/.bashrc с помощью любимого текстового редактора строку:

```
alias rm=»rm -i»
```

RMDIR

Есть два способа удаления директорий: можно поочередно удалить все содержимое директории, а потом использовать команду rmdir для удаления самой директории:

```
$mkdir mydir
$touch mydir/file1
$rm mydir/file1
$rmdir mydir
```

Этот способ обычно называют «метод удаления директорий для неудачников». Намного удобнее использовать команду 'rm -rf' для удаления директории со всем ее содержимым.

```
$rm -rf mydir
```

С осторожностью используйте эту команду, так как с ее помощью неопытному администратору (тем более с правами рута) очень легко наломать дров (и линукс-систем).

ИСПОЛЬЗОВАНИЕ WILDCARDS

ЧТО ТАКОЕ WILDCARDS

При повседневном использовании линукса часто возникают ситуации, когда нужно выполнить одну простую операцию (например rm) над множеством файлов. В этом случае не очень-то удобно перечислять все имена файлов в командной строке:

```
$ rm file1 file2 file3 file4 file5 file6 file7 file8
```

Решить эту проблему можно при помощи шаблонов замены (wildcards). Командный интерпретатор линукс поддерживает возможность указания множества файлов, используя шаблоны (по историческим причинам это еще называют «globbing»). Bash и другие команды линукс выбирают только те файлы, которые совпадают с шаблоном. Так, если вам нужно удалить файлы с file1 по file8, нужно написать:

```
$rm file[1-8]
```

А если нужно удалить все файлы, имена которых начинаются со слова file и файл с именем file:

```
$rm file*
```

Шаблон * соответствует любому символу, последовательности символов или «отсутствию символа». Конечно, шаблоны можно применять не только для удаления файлов, как будет показано ниже.

ЕСЛИ СОВПАДЕНИЕ НЕ НАЙДЕНО

Если вы хотите вывести список файлов в директории /etc/, имена которых начинаются с буквы «g» и файл с именем «g» (если такой существует), нужно написать:

```
$ls -d /etc/g*
/etc/gconf /etc/ggi /etc/gimp /etc/gnome /etc/gnome-vfs-mime-magic /etc/gpm /etc/group /etc/group-
```

Посмотрим что случится, если вы укажете шаблон, который не совпадает ни с одним именем файла:

```
$ls -d /usr/bin/asdf*jkl
ls: /usr/bin/asdf*jkl: No such file or directory
```

В этом примере мы попытались вывести список файлов, имена которых начинаются на «asdf» и заканчиваются на «jkl». Интерпретатор команд выдал сообщение, что файлов с такими именами не найдено.

СИНТАКСИС ШАБЛОНА: * И ?

Мы посмотрели, как работает глоббинг (подстановка имен файлов). А теперь рассмотрим подробнее синтаксис шаблонов:

* соответствует нулю или большему количеству символов:

- `/etc/g*` — все файлы в директории /etc/ , имена которых начинаются с «g» и файл с именем «g».
- `/tmp/my*1` — все файлы в директории /tmp, имена которых начинаются с «my» и заканчиваются на «1» (включая файл с именем «my1»)

? заменяет один любой символ:

- `myfile?` — любой файл, чье имя начинается со слова «myfile», за которым следует один любой символ.
- `/tmp/notes?txt` — соответствует файлам с именами «notes.txt» и «notes_txt» (если они существуют в /tmp/).

КВАДРАТНЫЕ СКОБКИ: []

Шаблон '[]' очень похож на '?', но позволяет явно указывать набор символов. Шаблон '[]' совпадает с одним символом из тех, что указаны в скобках. Также в скобках можно указать диапазон символов (для этого используется символ –/дефис) или несколько диапазонов подряд, тогда шаблон будет совпадать с одним любым символом из этого диапазона:

- `myfile[12]` — соответствует myfile1 и myfile2. Шаблон будет работать, пока существует хотя бы один из этих двух файлов.
- `[Cc]hange[Ll]og` — соответствует файлам с именами Changelog, ChangeLog, changeLog, и changelog. Как вы могли заметить, использование шаблона [] полезно при поиске имен, отличающихся регистром букв.

- `ls /etc/[0-9]*` — вывести список файлов в директории /etc/, имена которых начинаются с цифры.
- `ls /tmp/[A-Za-z]*` — вывести список файлов в директории /tmp/, имена которых начинаются с буквы (заглавной или прописной)

Конструкция [!] похожа на [], за исключением того, что она соответствует единичному символу, не упомянутому между [! и]. Например:

- `rm myfile[!9]` — удалит все файлы, имена которых состоят из слова «myfile» и идущей за ним одной цифрой, кроме файла «myfile9».

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Вот несколько примеров использования шаблонов. Так как bash интерпретирует символы ?, [,], * как шаблоны замены, необходимо принять меры предосторожности при использовании аргументов, содержащих эти символы. Например, если вы хотите создать файл, содержащий строку '[fo]*', то следующая команда сделает не то, что вы хотите:

```
$echo [fo]* > /tmp/mynewfile.txt
```

Если в вашей рабочей директории найдется один или несколько файлов, имена которых попадают под шаблон '[fo]*', то вы обнаружите в /tmp/mynewfile.txt список их имен, а не строку '[fo]*'. Но как же добиться того, чего мы хотели? Первый способ — это взять строку в одинарные кавычки. К строке в одинарных кавычках bash относится, как к обычной текстовой строке и не раскрывает символы замены.

```
$echo '[fo]*' > /tmp/mynewfile.txt
```

После выполнения этой команды, ваш файл будет содержать строку '[fo]*' ,как и ожидалось. Другой способ — заэкранировать спец.символы с помощью обратного слэша (\). Бэкслэш, стоящий перед спец.символом сообщает интерпретатору, что этот символ нужно рассматривать, как обычный текст, а не как шаблон.

```
$echo \[fo\]\* > /tmp/mynewfile.txt
```

Оба предложенных метода (одинарные кавычки и экранирование) дают желаемый результат. Раз уж мы заговорили об экранировании при помощи обратного слэша, стоит сказать, что чтобы указать текстовый символ «\» можно заключить его в одинарные кавычки или написать «\\» (эта комбинация будет воспринята интерпретатором, как обычный одинарный бэкслэш «\»)

Замечание: Двойные кавычки работают почти так же, как и одинарные, но позволяют bash-у интерпретировать некоторые спец. символы. Поэтому одинарные кавычки — лучший способ передать команде только текст. Для дополнительной информации о шаблонах читайте справку 'man 7 glob'. Для дополнительной информации об использовании кавычек, читайте раздел QUOTING справки 'man 8 glob'.

ЗАКЛЮЧЕНИЕ

Поздравляю, вы добрались до конца нашего обзора основ линукс! Надеюсь, материал оказался вам полезен. Темы, разобранные в этом пособии, включая основы bash, основные команды linux, ссылки и wildcards являются основой для следующей статьи об основах администрирования, в которой будет рассказано о регулярных выражениях, правах доступа, управлении аккаунтами пользователей и многом другом.

http://linuxgeeks.ru

Полезные команды терминала GRUB2

КРАТКО О GRUB

UB (GRand Unified Bootloader) – это самый популярный загрузчик операционной системы в мире пользователей *nix. В большинстве дистрибутивов по умолчанию используется GRUB версии 1 или новый – GRUB2. По сравнению с также достаточно популярным загрузчиком lilo, GRUB имеет больше возможностей, но за это его многие и не любят, считая его раздутым и переполненным ненужными функциями. Однако я считаю, что GRUB является прекрасным загрузчиком и он достоин серии статей о нём.

В данной статье я расскажу о некоторых полезных командах, которые вы можете использовать в терминале GRUB2. Мы не будем рассматривать их все, а рассмотрим самые нужные и важные. Для изучения полного списка команд вы можете использовать документацию grub, например, команда help выведет весь список доступных команд. Маловероятно, что вам понадобятся они все и поэтому в данной статье будут рассмотрены только самые привычные и полезные.

КАК ПОПАСТЬ В ТЕРМИНАЛ GRUB2?

Терминал GRUB2 – это строка ввода команд, которая доступна непосредственно из меню загрузчика GRUB2. Меню загрузчика – это список, состоящий из операционных систем, версий ядер и утилиты memtest, который отображается перед стартом непосредственно ОС. Для того, чтобы из меню загрузчика GRUB2 перейти в терминал, необходимо нажать клавишу «C». После этого у вас появится традиционное поле ввода команд.

ЕСЛИ МЕНЮ ЗАГРУЗЧИКА НЕ ВИДНО

Когда у вас установлен один дистрибутив, то, скорее всего, меню загрузчика будет скрыто. Для решения этой проблемы необходимо нажать клавишу «Esc» (escape) на первых порах старта компьютера.

Убедитесь, что в файле /etc/default/grub строка GRUB_HIDDEN_TIMEOUT равна положительному числу, а если это не так, то установите нужное значение в секундах, например:

```
GRUB_HIDDEN_TIMEOUT=10
```

КОМАНДЫ, ИМЕЮЩИЕ КОНКРЕТНОЕ ЗНАЧЕНИЕ

help – выводит список всех доступных команд.

help cat – в качестве опции можно использовать название конкретной команды для получения информации о ней.

root – важная команда, для вывода раздела жесткого диска, используемого в даный момет, например (hd0,1), и задания нового коренного (root) раздела.

Для изменения коренного раздела нужно использовать номер раздела (в специфическом формате GRUB), например:root (hd0,2). О том, как узнать номер раздела, читайте про команду ls.

ls – работает почти так, как и эта же команда в linux, но может показывать информацию о разделах.

Команда, в виде ls выведет список разделов жесткого диска (подобно fdisk –l, но представит разделы в виде формата записи для GRUB – (hdX,Y), где X и Y – числа).

ls раздел_диска (например, ls (hd0,1)) – выведет подробную информацию о конкретном раздела жесткого диска: UUID раздела и другую полезную информацию.

ls путь_до_папки (например, ls /) – полностью аналогично работе команды ls в терминале linux – выводит список файлов и папок в заданной папке. Примечательно, что вывод будет в формате через запятую (как при использовании опции –m в linux). К сожалению, вывод происходит без цвета (нет подсветки разным цветом разных типов файлов, как это можно сделать в linux опцией –color).

cat путь_до_файла – очень полезная команда, выводящая на экран содержимое текстовых файлов.

Пример использования:

```
cat/etc/default/grub
```

reboot – как не трудно догадаться по названию, перезагружает компьютер.

background_image путь_до_изображения – кошерная команда, позволяющая менять фоновое изображение загрузчика. Но довольно странно этим заниматься, ведь GRUB – это загрузчик, а не среда рабочего стола, но все же возможность есть.

linux путь_до_ядра опции_ядра и initrd путь_до_initrd_образа – загружает указанные ядро и initrd соответственно.

SET – универсальная команда для изменения параметров

set параметр=значение – используется для изменения параметров загрузчика. Например, в конфигурационном файле /boot/grub/grub.cfg есть строка set lang=en, которая задаёт параметр lang (язык). Для изменения этого параметра можно воспользоваться командой set:

```
set lang=ru
```

http://linuxnow.ru

Access Control List - списки контроля доступа

Поддерживаемые версии Ubuntu
Все с ядром 2.4.21 (?) и выше, а также другие OS

ВСТУПЛЕНИЕ

Итак, пришло время задуматься о безопасности вашей сети. В частности о назначении прав на каталоги и файлы для пользователей и групп. Стандартные права в операционных системах Unix не так гибки, как хотелось бы. К сожалению, они годятся для использования в простых схемах сети. Например, ситуацию, когда к одному и тому же каталогу нужно, чтобы несколько групп пользователей имели разные права доступа, не реализовать с использованием стандартных прав.

Для реализации сложных структур прав доступа используются расширенные права – ACL (Access control list - списки контроля доступа). Списки контроля доступом (ACL) дают большую гибкость, чем стандартный набор полномочий «пользователь/группа/остальные». ACL доступны в коммерческих Unix–системах, таких как IRIX или Solaris (и в Windows NT) в течение нескольких лет. В настоящее время, благодаря проекту TrustedBSD, ACL доступны в FreeBSD 5.0 и выше, а также в Linux. Возможность использования ACL позволяет администратору получить преимущество от использования более интеллектуальной модели безопасности.

Предполагается, что Вы уже знакомы со стандартными правами систем Unix. Если это не так, очень рекомендую сначала изучить документацию по ним. От того, насколько хорошо вы понимаете управление стандартными правами Unix, будет зависеть понимание материала этой статьи.

УСТОЯВШИЕСЯ ИСТИНЫ:

- Каждый пользователь входит в минимум одну группу. Группа, присваиваемая пользователю при его создании, называется основной. Все остальные группы, в которые будет включен пользователь, будут являться дополнительными. 1)
- Группа пользователей может содержать некоторое количество пользователей, но не может содержать или включаться в другие группы.
- Группа может быть пустой, т.е. не содержать в себе ни одного пользователя.

Чтобы добавить пользователя в ту или иную группу, достаточно отредактировать файл **/etc/group**:

```
syslog:x:103:
klog:x:104:
scanner:x:105:hplip, allexserv
nvram:x:106:
fuse:x:107:allexserv
```

Видно, что в листинге выше в группу scanner входят пользователи hplip и allexserv. Чтобы добавить в эту группу еще пользователей, просто перечислите их символьные имена через запятую.

СИНТАКСИС ФАЙЛА ПРОСТ:

имя_группы:пароль:GID:список_пользователей

Итак, основная мысль статьи – это использование расширенных прав ACL. 2)

ВКЛЮЧЕНИЕ ACL В СИСТЕМЕ

С какой-то там версии Ubuntu поддержка ACL уже включена и поддерживается ядром. 3) Но, как бы то ни было, по умолчанию, в системе, ACL не активированы. Для того, чтобы операционная система начала учитывать списки контроля, необходимо ей об этом сказать. Для этого потребуется отредактировать файл ***/etc/fstab*** и указать в нем, на каких разделах HDD следует учитывать ACL права. Проверить, поддерживает ли тот или иной раздел винчестера ACL, можно попытавшись установить эти самые ACL, командой ***setfacl:***

```
  /root > setfacl -m u:allexserv:rw-
,g:root:rw- qqq
  setfacl: qqq: Operation not supported
```

Operation not supported – операция не поддерживается – бодро рапортует вам система! Это признак того, что ACL на этом разделе винчестера, где лежит файл qqq не активированы. Что же, давайте отредактируем /etc/fstab и приведем его в должный вид:

```
  # /etc/fstab: static file system
information.
  #
  # <file system> <mount point>
<type>            <options>    <dump>
<pass>
  proc            /proc             proc
defaults    0     0

  # /dev/sda1
  UUID=1b09f304-1772-4a87-
bc1c-ee8c6170ef1e /     ext3
relatime,errors=remount-ro 0  1
  # /dev/sda5
  UUID=95b26917-535e-46ac-8d72-
443d46184bb5      /media/Profil      ext3
grpquota,acl,suid,dev,usrquota,relatime,ex
ec    0     2

  # /dev/sda6
  UUID=759a8adb-ed01-4d4f-b173-
9005ad165368      /media/work ext3  grpquo
ta,acl,suid,dev,usrquota,relatime,exec
0     2
```

```
  # /dev/sda7
  UUID=f90b3fb0-e515-4b4e-8a1b-
dfe7ed269a10 none swap  sw    0     0
  /dev/scd0 /media/cdrom0
udf,iso9660 user,noauto,exec,utf8 0 0
  /dev/fd0        /media/floppy0
auto  rw,user,noauto,exec,utf8 0    0
  /media/Work/test      /export/test
bind  bind  0
```

В тех разделах винчестера, в которых указан дополнительный параметр acl команды mount – grpquota,acl,suid - списки контроля будут поддерживаться в полном объеме. В моем случае поддержка ACL активирована на разделах ***/dev/sda5*** и ***/dev/sda6***.

После редактирования файла, лучше не мучиться и перезагрузить сервер, хотя, как утверждается в некоторых статьях, достаточно размонтировать раздел и смонтировать его вновь (такой номер у меня почему-то не прошел).

После вышеописанной операции ACL на соответствующих разделах готовы к работе.

УТИЛИТЫ ACL

Честно говоря, работа с ACL, на первый взгляд, может показаться сложной. Но если вникнуть в логику управления, то все становится на свои места. Учтите, что манипуляции с правами, особенно в сложных схемах, требуют хорошего логического мышления.

Существуют два типа ACL:

- ***ACL для доступа;***
- ***ACL по умолчанию.***

ACL для доступа – это список управления доступом для заданного файла или каталога. Проще говоря – это сами права на объект, которые будут контролировать доступ к этому объекту.

ACL по умолчанию – может быть связан только с каталогом, и, если файл в этом каталоге не имеет ACL для доступа, он использует правила, определённые в ACL по умолчанию, связанном с каталогом. ACL по умолчанию являются необязательными. ACL по умолчанию также можно сравнить с наследованием прав в, простите, Windows NT.

Управления ACL списками осуществляется всего лишь двумя командами: 4)

- ***setfacl*** – используется для назначения, модификации и удаления ACL прав.
- ***getfacl*** – используется для просмотра установленных ACL.

Несколько слов о взаимодействии других команд (такие как копирование, перемещение, архивирование и т.п.) с правами ACL. Вот выдержка из статьи:

К сожалению, большинство Unix-утилит все еще не поддерживает ACL. Например, tar не архивирует и не восстанавливает ACL, в FreeBSD NFS также игнорирует их. Ни формат файлов в утилите tar, ни протокол NFS не имеет ни намека на возможность использования ACL. Тем не менее, архивы полного раздела UFS1, сделанные с помощью tar или dump, восстанавливают каталог .attribute, и утилита dump в FreeBSD модифицирована для «понимания» UFS2 (включающую ACL). Каталог-скелет archivers/star поддерживает ACL. Вы даже можете работать с архивами, созданными в Linux и FreeBSD С помощью star и предохраняющей расширенные атрибуты (включающие и ACL).

Но не все так грустно! Перевод статьи был написан в 2006 году (ксожалению до оригинала я так и не добрался). В другой же статье, более поздней, сказано:

Ядро Red Hat Enterprise Linux 4 обеспечивает поддержку ACL для файловой системы ext3 и экспортируемых файловых систем NFS. Списки ACL также работают в файловых системах ext3, доступных через Samba. … Команды cp и mv копируют и перемещают все списки ACL, связанные с файлами и каталогами.

Хоть большинство стандартных команд, призванных производить операции над файлами, уже поддерживают ACL то, например, команды архивирования tar и dump ACL не архивируют. Для архивации данных с установленными ACL используется программа star. Вкратце ее рассмотрим позже. Замечу, что для NFS ACL поддерживаются уже по умолчанию. Вообще, успешное использование ACL зависит от поддержки ACL файловой системой и поддержки ACL операционной системой на клиентской машине.

Итак, рассмотрим синтаксис и параметры ***getfacl*** и ***setfacl***.

УТИЛИТА GETFACL

О getfacl сильно и говорить нечего. Она выводит листинг ACL прав для указанных объектов.

Примеры использования:

- ***getfacl**** – отобразит права ACL для всех объектов в текущем каталоге;
- ***getfacl sobaka.txt*** – отобразит ACL для файла sobaka.txt;
- ***getfacl kartosh**** – отобразит ACL для всех файлов в текущем каталоге, которые начинаются на kartosh;
- ***getfacl -R **** – отобразит ACL для всех объектов (включая подкаталоги и их содержимое) текущего каталога.

Чтобы посмотреть, установлены ли ACL на объектах, достаточно воспользоваться командой ls -l:

Символ «+» в конце списка стандартных прав сообщает о наличии установленных прав ACL:

```
  root@sytserver:/media/Work/test# ls -l
  итого 28
  drwxrwxrwt      2 root root       4096
2009-07-24 21:20 allex
  -rwxr-x---+     1 root root       19
2009-07-25 14:45 qwert
```

Теперь рассмотрим, что же отобразит команда getfacl:

```
  root@sytserver:/media/Work/test# getfacl
qwert
  #file: qwert
  #owner: root
  #group: root
  user::rwx
  user:child:rw-
  group::r--
  mask::rw-
  other::---
```

Из примера видно, что без использования ACL, пользователь child не получил бы права к файлу qwert, т.к. не входит в группу root и не является владельцем файла. Права ACL, в данном случае, предоставили ему права на чтение и запись этого файла. Давайте подробнее разберем листинг getfacl:

#file: qwert – Имя файла

#owner: root – Владелец файла (основные права Unix)
#group: root – Группа файла (основные права Unix)
user::rwx – Права для владельца файла (основные права Unix)
user:child:rw- – Права ACL для пользователя child
group::r-- – Права для группы файла (основные права Unix)
mask::rw- – Эффективная маска
other::--- – Права для пользователя «все остальные»
Что касается ключей getfacl, то есть пара из них, которые стоит рассмотреть, но рассматривать их будем тогда, когда будем изучать setfacl.

УТИЛИТА SETFACL

Теперь об утилите setfacl. Как уже говорилось выше, утилита setfacl предназначена для установки, модификации или удаления ACL.

Списки ACL можно задать:

- На уровне пользователей – назначаются ACL конкретным пользователям;
- На уровне групп – назначаются ACL конкретным группам;
- С помощью маски эффективных прав – ограничение максимальных прав для пользователей и/или групп;
- Для пользователей, не включённых в группу данного файла – это т.н. пользователь «Все остальные»;

Рассмотрим простой синтаксис setfacl:
setfacl <опции> <ключ> <список правил> <объект>

- <опции> – задает дополнительные опции;
- <ключ> – задает режим работы утилиты;
- <список правил> – собственно, сами правила доступа к объекту;
- <объект> – объект, к которому применяется ACL, в большинстве случаев это файл или каталог.

Часто используемые ключи:

КЛЮЧ	ОПИСАНИЕ
–set или –set file*	– Устанавливает новые указанные права ACL, удаляя все существующие. Необходимо, чтобы наравне с задаваемыми правилами ACL были также указаны стандартные права Unix, в противном случае будет давать ошибку;
–m или –M file*	– Модифицирует указанные ACL на объекте. Другие существующие ACL сохраняются.
–x или –X file*	– Удаляет указанные ACL права с объекта. Стандартные права Unix не изменяются.

* – При использовании –M, – –set, –X – разрешения будут браться из указанного файла file, который должен быть заранее подготовлен в формате вывода ACL разрешений командой getfacl. Подготовить файл можно либо вручную, в формате вывода getfacl, либо использовать команду getfacl –R file > file_out. Где file это файл(ы) или каталог(и) с которых нужно снять ACL, а file_out – текстовый файл куда запишутся снятые ACLправа.

Часто используемые опции:

ОПЦИЯ	ОПИСАНИЕ
–b	– Удаляет все ACL права с объекта, сохраняя основные права;
–k	– Удаляет с объекта ACL по умолчанию. Если таковых на объекте нет, предупреждение об этом выдаваться не будет;.
–d	– Устанавливает ACL по умолчанию на объект.
–restore= file	– Восстанавливает ACL права на объекты из ранее созданного файла с правами. 5)
–R	– Рекурсивное назначение (удаление) прав, тобишь пройтись по всем подкаталогам.

* <perms> – Сами правила для пользователя или группы. Могут принимать значения (r), (x), (w), или сочетания друг с другом.

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Рассмотрим различные примеры использования назначения, модификации и удаления ACL. Выше мы уже выводили листинг, при помощи команды getfacl для файла qwert, на котором уже установлены ACL для пользователя child:

```
root@sytserver:/media/Work/test# getfacl qwert
# file: qwert
# owner: root
# group: root
user::rwx
user:child:rw-
group::r-
mask::rw-
other::--
```

Теперь давайте добавим к этому файлу еще пользователя allexserv:

```
root@sytserver:/media/Work/test# setfacl -m u:allexserv:rwx qwert
root@sytserver:/media/Work/test# getfacl qwert
# file: qwert
# owner: root
# group: root
user::rwx
user:allexserv:rwx
user:child:rw-
group::r-
mask::rwx
other::--
```

Обратите внимание, как изменилась эффективная маска. Все ACL права сложились: у пользователя child rw– и у пользователя allexserv rwx. Но маска вовсе не разрешает пользователю child иметь право на выполнение (x) файла qwert. Посмотрите, что произойдет, если принудительно изменить эффективную маску:

```
root@sytserver:/media/Work/test# setfacl -m m:r qwert
```

Формирование списка правил:

ОПЦИЯ	ОПИСАНИЕ	ПРИМЕР ИСПОЛЬЗОВАНИЯ
u:<uid>:<perms>*	– Назначает ACL для доступа заданному пользователю. Здесь можно указать имя или UID пользователя. Это может быть любой пользователь, допустимый в данной системе.	***Пример:*** ***setfacl –m u:allexserv:rw myfile.odt***
g:<gid>:<perms>*	– Назначает ACL для доступа заданной группе. Здесь можно указать имя или GID группы. Это может быть любая группа, допустимая в данной системе.	***Пример:*** ***setfacl –m g:children:r myfile.odt***
m:<perms>*	– Назначает маску эффективных прав. 6)	***Пример:*** ***setfacl –m m:rx myfile.odt***
o:<perms>*	– Назначает ACL для доступа пользователям, не включённым в группу файла. Это пользователь «все остальные», как в стандартных правах Unix.	***Пример:*** ***setfacl –m o: myfile.odt*** ***– убирает все права (отсутствие прав)***

```
root@sytserver:/media/Work/test# getfacl qwert
# file: qwert
# owner: root
# group: root
user::rwx
user:allexserv:rwx       #effective:r-
user:child:rw-           #effective:r-
group::r-
mask::r-
other::--
```

Теоретически, как гласит документация, в данном случае пользователь child не может удалить файл (про allexserv ничего не говорю, т.к. он входит в группу root у меня). Но проведя тест, пользователь child все–таки удалил файл, правда перед удалением система спросила:

```
child@sytserver:/media/Work/test$ rm qwert
rm: удалить защищенный от записи обычный файл `qwert'?
```

В то же время попробовал отредактировать файл qwert под пользователем child и попытался записать изменения. Здесь система в записи отказала. Вобщем очень странный момент.

Теперь давайте удалим с файла qwert права ACL для пользователя allexserv:

```
root@sytserver:/media/Work/test# setfacl -x u:allexserv qwert
root@sytserver:/media/Work/test# getfacl qwert
# file: qwert
# owner: root
# group: root
user::rwx
user:child:rw-           #effective:r-
group::r-
mask::r-
other::--
```

Чтобы удалить все ACL права с файла можно воспользоваться командой: ***setfacl –b qwert***

Очевидно, что таким образом можно назначать и удалять ACL права для пользователей и групп на файлы и каталоги.

Давайте еще рассмотрим т.н. наследование прав. Если не задано иное, то все объекты, создаваемые в каталоге, у которого есть ACL, не наследуют права ACL с каталога, в котором они создаются. Но иногда возникают ситуации, когда необходимо, чтобы все объекты, создаваемые внутри каталога наследовали его ACL права. Это называются ACL по умолчанию.

Задача: создадим каталог ***Proverka*** и назначим ему владельца child и группу children (разумеется, пользователь и группа должны существовать в системе). Установим ACL права для пользователя allexserv и пользователя mysql. Установим ACL по умолчанию на каталог Proverka так, чтобы создаваемым объектам внутри него также назначались ACL.

Создаем каталог, устанавливаем права и владельца:

```
root@sytserver:/media/Work/test# mkdir Proverka
root@sytserver:/media/Work/test# chmod 700 Proverka
root@sytserver:/media/Work/test# chown child:children Proverka
root@sytserver:/media/Work/test# ls -l
итого 24
drwxrwxrwt 2 root  root     4096 2009-07-24 21:20 allex
drwx--- 2 child children 4096 2009-07-25 23:36 Proverka
```

Назначаем ACL права сразу для двух пользователей, перечислив их через запятую:

```
root@sytserver:/media/Work/test# setfacl -m u:allexserv:rwx,u:mysql:rwx Proverka
root@sytserver:/media/Work/test# getfacl Proverka
# file: Proverka
# owner: child
# group: children
user::rwx
user:mysql:rwx
user:allexserv:rwx
group::--
mask::rwx
other::--
```

Теперь назначим ACL по умолчанию. При назначении ACL по умолчанию также требуется в обязательном порядке указывать и основные права в виде u::rwx,g::–,o::– где u – user – владелец, g – group – группа, o – other – все остальные:

```
  root@sytserver:/media/Work/
test# setfacl -d -m u::rwx,g::-,o::-
,u:allexserv:rwx,u:mysql:rwx Proverka
  root@sytserver:/media/Work/test# getfacl
Proverka
  # file: Proverka
  # owner: child
  # group: children
  user::rwx
  user:mysql:rwx
  user:allexserv:rwx
  group::--
  mask::rwx
  other::--
  default:user::rwx
  default:user:mysql:rwx
  default:user:allexserv:rwx
  default:group::--
  default:mask::rwx
  default:other::--
```

Видно, что появились строки, начинающиеся с default. Это и есть права по умолчанию, которые будут принимать все создаваемые внутри объекты. Проверим, создав пустой файл myfile.txt и подкаталог MyKatalog в каталоге Proverka:

```
  root@sytserver:/media/Work/test/
Proverka# touch myfile.txt
  root@sytserver:/media/Work/test/
Proverka# mkdir MyKatalog
  root@sytserver:/media/Work/test/
Proverka# getfacl *
  # file: myfile.txt
  # owner: root
  # group: root
  user::rw-
  user:mysql:rwx        #effective:rw-
  user:allexserv:rwx    #effective:rw-
  group::--
  mask::rw-
  other::--

  # file: MyKatalog
  # owner: root
  # group: root
  user::rwx
  user:mysql:rwx
  user:allexserv:rwx
  group::--
  mask::rwx
  other::--
  default:user::rwx
  default:user:mysql:rwx
  default:user:allexserv:rwx
  default:group::--
  default:mask::rwx
  default:other::--
```

Обратите внимание, что каталог MyKatalog не только приобрел ACL, но и также приобрел и ACL по умолчанию. Т.е. те объекты, которые будут создаваться в подкаталоге MyKatalog, тоже будут наследовать ACL по умолчанию.

Удалить права по умолчанию можно: setfacl –k Proverka.

Если нужно также удалить права по умолчанию и в подкаталогах, то добавьте ключ –R (рекурсия):

```
setfacl -R -k /media/Work/test/Proverka
```

Здесь мы оперировали двумя пользователями. Но ничто не мешает вам оперировать также целыми группами пользователей.

АВТОМАТИЧЕСКИЕ ОПЕРАЦИИ

Любой администратор стремится к оптимизации. Понятно, что назначить вручную 100 объектам одни и те же права – нудное занятие и нецелесообразное. Есть некоторые фишечки, которые могут облегчить подобные задачи.

Копирование ACL прав с одного объекта на другой.

Принцип прост:

- Снять ACL с одного объекта
- Записать их на другой объект

В документации приведен следующий пример:

```
getfacl file1 | setfacl -set-file=- file2
```

Где file1 – объект, с которого снимаем ACL командой getfacl, а далее устанавливаем ACL командой setfacl на объект file2. Обратите внимание на запись – –set–file=– в конце указан символ »–», который берёт информацию от команды getfacl (со стандартного вывода).

Справедлива будет также такая запись:

```
getfacl file1 | setfacl -M- file2
```

В этом случае права на file2 не заменяются, как при использовании – –set, а добавляются к уже существующим правам ACL.

Копирование прав ACL каталога в права по умолчанию этого же каталога

```
getfacl -access dir | setfacl -d -M- dir
```

В этом примере getfacl получает все права, которые вы установили на каталог dir и устанавливает их на этот же каталог dir, делая их правами по умолчанию. Очень удобно. Обратите внимание на ключ – –access у команды getfacl. При вызове getfacl без параметров, она отображает все права ACL, включая права по умолчанию. Здесь же, ключ – –access заставляет getfacl показать только права ACL на каталог, а права по умолчанию (если таковые имеются у каталога) – скрыть. Обратный ключ – это ключ –d:

```
getfacl -d Proverka
```

заставит getfacl показать только права по умолчанию, а права ACL на сам каталог – скрыть. Кстати, в нашем примере, с каталогом Proverka, на который мы назначали права по умолчанию, чтобы не писать строку:

```
setfacl -d -m u::rwx,g::-,o::-
,u:allexserv:rwx,u:mysql:rwx Proverka
```

можно было бы воспользоваться именно этой фишечкой, как то так:

```
getfacl -access Proverka | setfacl -d
-M- Proverka
```

Результат был бы тот же.

Вот некоторые часто используемые опции star:

ОПЦИЯ	ОПИСАНИЕ
–c	Создаёт файл архива
–n	Отключает извлечение файлов, используется в сочетании с –x для просмотра списка извлекаемых файлов
–r	Заменяет файлы в архиве. Файлы записываются в конец архива, заменяя любые файлы с тем же путём и именем
–t	Выводит содержимое файла архива
–u	Обновляет файл архива. Файлы записываются в конец архива, если их ещё не было в архиве или если они новее, чем файлы с тем же именем в архиве.7)
–x	Извлекает файлы из архива. Если используется с ключом –U и файл в архиве старее, чем соответствующий файл в файловой системе, такой файл не извлекается
–help	Выводит наиболее важные параметры
–xhelp	Выводит менее важные параметры
–/	Оставляет ведущую косую черту в имени файла при извлечении файлов из архива. По умолчанию она убирается
–acl	При создании архива или извлечении файлов, архивирует или восстанавливает все ACL, связанные с файлами или каталогами

Вот и всё. А вообще, не поленитесь почитать man getfacl – очень занимательно!

ОПЕРАЦИИ НАД ОБЪЕКТАМИ С ACL

В заключении хочу еще раз обратить внимание на операции с объектами, у которых установлены ACL, такие как копирование, перемещение, архивирование. Выше мы уже упоминали о них. Некоторые замечания:

- При перемещении (mv) никаких дополнительных параметров не нужно;
- При копировании (cp) необходимо использовать ключ –p, в противном случае ACL права будут потеряны;

Внимание!!!

По умолчанию графический интерфейс при копировании не учитывает ACL права!

- При архивировании или распаковке вместо tar используйте утилиту star.

Утилиту star нужно будет установить из репозиториев:

```
apt-get install star
```

Пример для архивирования утилитой star с сжатием:

```
star –czv –Hexustar –acl –f /tmp/homedir.tgz /media/Profil/home
```

Пример для разархивирования в текущий каталог:

```
star –xv –Hexustar –acl –f homedir.tgz
```

Вот собственно и все, что я хотел рассказать про ACL. Применяйте логику и смекалку – и все будет хорошо.

Администрирование, система, Права доступа, HOWTO

1) Наличие дополнительных групп может также облегчить написание правил монтирования ресурсов при входе в систему, например, с использованием модуля pam_mount. Почитать можно здесь.

2) Часть информации взята с этого www.bsdportal.ruперевода. Также очень хорошо описан синтаксис ACL здесь www.rhd.ru. Надеюсь, авторы будут не против того, если часть текста я позаимствую из этих статей.

3) Обратите внимание, что в других операционных системах Unix, например, FreeBSD 5.0 и выше, потребуются некоторые дополнительные действия по включению ACL. Также, возможно, вам потребуется установить пакет acl, использовав следующую команду: sudo apt–get install acl.

4) Помимо консольных команд, есть также графическая оболочка для установки ACL прав. Приложение называется eiciel, ее описание можно найти, например, здесь www.linuxcenter.ru. По–моему, по умолчанию оно не ставится в системе, так что ставить ее придется из репозиториев. Ввиду того, что эта статья ориентирована на системных администраторов, рассмотрения графической утилиты здесь не состоится.

5) Полезно для отката разрешений. Разумеется, необходимо сначала сделать резервную копию разрешений в текстовый файл file. Сделать можно либо вручную, в формате вывода getfacl, либо использовать команду getfacl –R file > file_out. Где file это файл(ы) или каталоги, с которых нужно снять ACL, а file_out – текстовый файл, куда запишутся снятые ACL права.

6) Маска – это объединение всех разрешений группы–владельца и всех записей пользователей и групп. Маска задает максимальные права доступа для всех пользователей, за исключением хозяина и групп. Установка маски представляет собой самый быстрый путь изменить фактические (эффективные) права доступа всех пользователей и групп. Например, маска (r – –) показывает, что пользователи и группы не могут иметь больших прав, чем просто чтение, даже если им назначены права доступа на чтение и запись. Например, если на файл koshka назначили ACL пользователю allexserv с правами (r w x), а эффективную маску выставили в (r x), то пользователь лишается права (w), не смотря на то, что по ACL он имеет это право.

7) Этот параметр работает только, если архив представляет собой файл или незаблокированную ленту, которую можно стирать

Соловьев Алексей
http://help.ubuntu.ru

Communigate: Часть 3

Настройка почтового домена, учетных записей пользователей

ВВЕДЕНИЕ

В первой и второй частях цикла о Communigate (CommuniGate Pro) были рассмотрены вопросы общей организации и назначения интеграционного программного комплекса Communigate.

Мы изучили основные компоненты, образующие логический состав сервера и составляющие его структуру.

Описали нюансы, возникающие при выборе различных платформ для развертывания Communigate.

Кроме того, в статьях были подробно рассмотрены вопросы установки и базовой настройки Communigate, достаточные для ввода программы в эксплуатацию и начала работы с ней. В процессе описания этих вопросов предпринята попытка объединить изложенные рекомендации для версии 5.0.5 Communigate и более поздних.

В третьей части цикла мы разберём настройку почтового домена и учетных записей пользователей, а также процесс работы почтового модуля.

НАСТРОЙКА ПОЧТОВОГО ДОМЕНА

Для предварительной настройки на сервере Communigate почтового домена необходимо внести ряд общесистемных настроек. Для этого нужно зайти на сервер под учетной записью postmaster и на вкладке General в соответствующих разделах задать требуемые параметры. К их числу относятся: Main Domain Name (главное имя домена), Server Internals Log – задает возможность выбора уровня логгирования событий, Crash Recovery – режим восстановления после сбоев, Server Time – время на сервере, Server Up–Time – время работы сервера от момента последнего выключения (перезагрузки), Server OS – тип операционной системы на сервере, используемая версия сервера (в описываемом примере 5.0.5 и выше), MAPI Version – версия MAPI, Server IP Address(es) – назначенные Ip адреса сервера, Name Server(s) IP Address(es) – [127.0.0.1]. задано как localhost. Затем на вкладке Intercept кажется логичным задать псевдоним имени для postmaster в целях пересылки лог–файлов на другой почтовый аккаунт для сбора статистики.

После чего в подразделе Network раздела Settings нужно задать LAN IPs – перечень или список ip адресов сети, которая для сервера является локальной.

Затем можно перейти непосредственно к настройке домена, который будет обслуживать Communigate, – это вкладка Domains. Раскрыв ее, можно увидеть интерфейс настройки домена. На подвкладке Domains есть возможность создать новый или первый домен, который будет обслуживаться Communigate. Для этого нужно щелкнуть по “Create Domain”, предварительно задав имя нового домена справа, в свободном поле, находящемся на одном уровне с кнопкой “Create Domain”. После подтверждения выполнения выбранных действий домен создастся. И его имя можно будет наблюдать в нижнем поле. Здесь он может быть «раскрыт» для выявления его изменяемых параметров и внесения в них требуемых данных. Но пока что щелчком мыши раскроем вверху следующее подменю “Domain Default”. Здесь приведены ряд параметров, которые регулируют настройки по умолчанию, изменяющие поведение Communigate при работе с выбранным доменом. Это выбор уровня логгирования событий, связанных с аккаунтами этого сервера Account Log, Mailbox Log – тоже, но для почтовых ящиков. Затем предлагается к рассмотрению поведение сервера при использовании механизмов внешней аутентификации – Consult External Authenticator, при обработке почтовых сообщений для неизвестных имен – «Mail to Unknown Names is», и поведение сигналов для неизвестных имен – «Signals to Unknown Names are:» и целый ряд дополнительных настроек. Рассмотрим наиболее важные из них. Здесь, на этой вкладке, необходимо активировать те сервисы, которые будут доступны для данного домена. В первую очередь это сервисы, которые фигурируют в подразделе Enabled Services. К их числу относятся Mail, FTP, POP, MAPI, IMAP, TLS, PWD, S/MIME, ACAP/LDAP, WebMail, WebCal, WebSite, RADIUS, RELAY, SIP, Mobile, PBX. В данном случае можно предпринять попытку поэтапного осмысления всех возможностей путем их последовательного изучения в действии. Это, как по методике написания правил firewall – сначала для домена на вкладке “Domain Default” отметим все галочками (т.е. все активируем), затем на соседней вкладке “Account Default” по очереди активировать только то, что кажется вам необходимым для работы ваших аккаунтов. Результат применится только к вашим аккаунтам в усеченном, вами выбранном виде, а для домена в целом, в тоже время, будет задействован режим «наибольшего благоприятствования», то есть без ограничений функционала. Манипулируя, таким образом, настройками можно точнее понять, для чего что предназначено и более точно выверять свои настройки. Это же касается и параметров ниже на этой же странице – в подразделе «Resources». Для параметров Accounts, Mailing Lists, RPOP Accounts, MAPI Connections оставляем значения unlimited, а на вкладке “Account Default” усекаем те или иные «излишества» по своему усмотрению. Конечно, у каждого инженера может быть своя тактика в этих вопросах и мое мнение на этот счет не является абсолютно окончательным и единственно верным.

НАСТРОЙКА УЧЕТНЫХ ЗАПИСЕЙ

На вкладке “Account” в основном меню левой колонки административного веб–интерфейса можно завести новых пользователей в систему, задать им нужные настройки и режимы работы. Для заведения нового пользователя нужно в пустой графе справа от кнопки “Create account” завести учетную запись для этого пользователя и создать его с помощью одноименной кнопки. После этих манипуляций открывается страница “Settings” для создаваемого аккаунта.

Здесь нужно задать ему настоящее имя в графе Real Name, в графе "CommuniGate Password" задать его пароль. Ниже, в разделе "Authentication" задать параметры для аутентификации, которые подразбиты на четыре подраздела. Это "CommuniGate Password" – позволяет использовать пароль вообще, изменять его пользователем и использовать механизмы шифрования пароля. Затем в подразделе "Server OS Integration" можно объединить использование в Communigate системных учетных записей (лично я не рекомендую этого делать из соображений безопасности), использовать возможность внешних механизмов аутентификации – "External Authentication" и использовать протокол Kerberos.

Ниже, в разделе "Enabled Services" стоит перечень имеющихся сервисов для вновь заводимого пользователя Communigate, причем для него уже выставлены настройки по умолчанию. Если нужно изменить в силу тех или иных причин умолчальные настройки для какого–то пользователя, то нужно снять «галочку» с "default" и нажать на кнопку "update". В этом случае применятся все настройки, деланные нами ранее и разблокируются «не умолчальные» настройки для текущей учетной записи. Надо отметить, что каждое нажатие кнопки "update" приводит к сохранению вводимой информации и ее фиксации. При удачном применении update, одноименная надпись зеленого цвета появится под теми разделами, которые подверглись изменению.

Таким образом, сняв настройки по умолчанию с вводимого пользователя, разблокируется весь спектр сервисов, которые для него доступны. К их числу относятся Mail, FTP, POP, MAPI, IMAP, TLS, PWD, S/MIME, ACAP/LDAP, WebMail, WebCal, WebSite, RADIUS, RELAY, SIP, Mobile, PBX. Отметив «галочками» требуемое и сняв ненужное, можно добиться желаемого уровня функционала применительно к выбранному пользователю.

Ниже, в позициях "Limits" и " Mail Quota Processing" задаются ограничения на использование файловой системы и порядок уведомления пользователя, нарушающего эти ограничения в силу тех или иных обстоятельств. Здесь "Mail Storage" – подразумевает общий объем принимаемых и хранимых сообщений, "File Storage" – ограничивает максимальный суммарный размер всех файлов персонального сайта пользователя, "Mailboxes" – ограничивает максимальное число почтовых ящиков, которые могут быть созданы для данного пользователя, "Files" – определяет максимально возможное число файлов персональных веб–страничек.

Немного возвращаясь назад, надо отметить, что все настройки по умолчанию для пользователей домена можно найти на вкладке "Domains" –> "Account Defaults". Здесь они могут быть видоизменены и в созданном качестве применены по умолчанию ко всем, вновь создаваемым учетным записям.

ОСНОВНЫЕ ПРИНЦИПЫ РАБОТЫ ПОЧТЫ В COMMUNIGATE

Одной из основных функций Communigate является передача сообщений электронной почты. Функционируя как MTA (агент передаче почты), сервер получает сообщения из различных источников (модули, внутренние компоненты и т.д.) и доставляет эти сообщения в удалённые или локальные почтовые ящики, задействуя свои имеющиеся модули. Все, полученные сервером, сообщения хранятся как отдельные файлы в директории Queue, находящейся внутри «директории данных» Communigate. Каждое сообщение может быть поставлено в несколько разных очередей на доставку (если оно имеет несколько получателей). Каждый модуль коммуникации может обслуживать одну или несколько очередей. К примеру, SMTP модуль обслуживает одну очередь для каждого домена в Интернет. Сервер Communigate может автоматически обрабатывать сообщения, используя автоматические правила. Общие для сервера и общие для кластера правила применяются ко всем сообщениям, поступаемым как в сервер, так и в кластер. Эти правила применяются компонентом «постановка в очередь» до того, как сообщение будет поставлено в очередь передающего модуля. Направляя сообщение пользователю сервера Communigate, модуль местной доставки применяет правила для этого пользователя. Эти правила являются правилами, заданными для определённого пользователя и применяются наряду с правилами, заданными для домена в целом. Причем, как правило, доменные правила назначаются первыми и имеют более охватывающую структуру.

ФИЛЬТРЫ СООБЩЕНИЙ

Фильтры Communigate являются намного более надёжным решением, чем различные отдельные «почтовые сканеры» для SMTP:

- Отдельные «сканеры»– ретрансляторы SMTP обычно обладают только базовыми функциями SMTP. Так как все SMTP соединения должны проходить через эти ретрансляторы, а не через SMTP модуль Communigate, то расширенные функциональные возможности SMTP модуля Communigate не используются пользователями и другими SMTP серверами.
- Отдельные «сканеры»– ретрансляторы SMTP обычно обладают меньшей, по сравнению с Communigate, производительностью и надежностью работы. Если «сканер» – ретранслятор прекращает работу, то SMTP функции Communigate также становятся недоступными.
- Отдельные «сканеры»– ретрансляторы SMTP обычно не умеют сканировать несколько сообщений одновременно, так что при сканировании большого сообщения SMTP трафик на сервере просто останавливается.
- Отдельные «сканеры не умеют сканировать сообщения, поставленные на сервер не через SMTP. Например, сообщение, созданное через веб–интерфейс пользователя и направленное пользователю этого же сервера, будет доставлено без требуемого сканирования.

Внешние фильтры работают совместно с Communigate. Они не используют в своей работе протоколы передачи сообщений. Вместо этого, сервер передаёт им файл с сообщением до того, как сообщение ставится в очередь какого–либо модуля. В результате, будут сканироваться все сообщения, а не только те сообщения, которые были отправлены через определённый протокол передачи почты. Компонент «постановка в очередь» настраивается на использование нескольких обработчиков (нитей), и, таким образом, одновременно могут сканироваться несколько сообщений. В результате длинные сообщения, сканирование которых занимает несколько секунд, не останавливают поток сообщений в

целом. Для дополнительных модулей сторонних производителей, распространяемых компанией "CommuniGate Systems", обычно требуются дополнительные лицензионные ключи. Сейчас доступно несколько дополнительных модулей, которые можно выбрать на сайте.

РАБОТА SMTP МОДУЛЯ

В SMTP модуле Communigate реализована передача сообщений электронной почты, используя протоколы SMTP и ESMPT через TCP/IP. Простой протокол передачи почты позволяет компьютерам передавать сообщения через имеющиеся сетевые соединения. Компьютер, который имеет сообщение для отправки, соединяется с компьютером получателя и устанавливает сетевое соединение. Затем он отправляет одно или несколько сообщений и закрывает сетевое соединение. Почтовые приложения используют протокол SMTP для передачи сообщений на почтовые сервера, а почтовые сервера пересылают поступившие сообщения получателям. Все почтовые программы имеют настройку, указывающую сетевой адрес компьютера, выступающего в роли почтового сервера. Когда у почтовых приложений есть сообщение для отправки, они открывают TCP соединение на этот адрес и предпринимают попытку отправки сообщения. SMTP модуль Communigate поддерживает специальные «безопасные порты» и расширение SMTP STARTTLS и может получать и отправлять почту через безопасные (зашифрованные) соединения. SMTP модуль Communigate поддерживает расширение AUTH и позволяет удалённым пользователям аутентифицироваться до передачи сообщений. В SMTP модуле Communigate также реализована разновидность протокола, называемая LMTP (протокол передачи локальной почты), используемая для локальной связи.

МЕСТНАЯ ДОСТАВКА

Модуль местной доставки обрабатывает сообщения, предназначенные для внутренних пользователей сервера. При этом используется менеджер папок для сохранения сообщений в папках пользователя. Модуль местной доставки применяет для автоматической обработки почты ко всем сообщениям, направленным для местного (локального по отношению к серверу) пользователя. Эти правила могут указать на сохранение сообщения в другой папке, перенаправление сообщения на другой адрес и т.д. После того, как сообщение сохранено в папке пользователя, к нему может быть получен доступ через любой из имеющихся модулей для доступа к сообщениям. Модуль местной доставки может поддерживать прямую адресацию папок и детализированные адреса пользователя. Он может ограничить число сообщений, которое пользователь получает в течении указанного периода времени. Эта возможность позволяет серверу минимизировать ущерб от возможного зацикливания почты.

RPOP МОДУЛЬ

В RPOP модуле Communigate реализована функция получения сообщений электронной почты по протоколу POP3 поверх TCP/IP сети. Этот модуль позволяет пользователям Communigate забирать почту из своих папок на сервере, а RPOP модуль забирает сообщения с других (удалённых) хостов и доставляет их в папки пользователей или в другие места. Для каждого зарегистрированного пользователя, RPOP модуль может забирать сообщения из нескольких удалённых папок. RPOP модуль может забирать почту для всего домена, используя метод «общих пользователей домена» и раздавать забранные сообщения получателям. RPOP модуль поддерживает нестандартные MSN POP3 сервера: если имя удалённого хоста оканчивается на .msn.com, то модуль использует нестандартный метод AUTH MSN для входа на этот сервер.

ВЫВОДЫ

В статье рассмотрены основные вопросы, связанные с конфигурированием почтового домена и ввода в работу пользовательских аккаунтов Comminigate. В следующей статье цикла речь пойдет о настройке веб–интерфейса пользователей.

Александр Деревянко
www.ibm.com

Быстрое восстановление GRUB 2

GRUB в версии 2 претерпел очень много изменений. Теперь стандартные пути восстановления загрузчика не подойдут и придётся повозиться.

Необходимо загрузиться в liveCD дистрибутива и зайти в терминал.

Далее вводим команду (от суперпользователя):

```
fdisk -l
```

Эта команда выведет все сектора жесткого диска, из которого мы сможем узнать, где именно установлена наша linux.

Теперь необходимо примонтировать разделы с установленной системы для дальнейшей работы. Из вывода предыдущей команды мы получили информацию о разметке, и теперь мы точно знаем, что нужно примонтировать.

Например, если linux установлен на sda1, то выполним (от суперпользователя):

```
mount /dev/sda1 /system
mount –bind /dev /
system/dev
mount –bind /proc /
system/proc
```

Если у вас есть дополнительные каталоги, то примонтируйте их тоже.

Теперь нам нужно использовать наш примонтированный раздел, как коренной (от суперпользователя):

```
chroot /system
```

Теперь мы можем установить GRUB 2 в первичный сектор командой:

```
grub-install /dev/sda
```

Если возникла ошибка:

```
grub-install –recheck /
dev/sda
```

Можно перезагрузить компьютер. В большинстве случаев эта инструкция должна сработать на 100%, меня она подводила только один раз (потому что ситуация была сложная).

http://linuxnow.ru

urpmi:
краткое руководство пользователя

Давным-давно, когда еще весь цивилизованный мир не знал про пакетные менеджеры, установка ПО в Linux была длительная и мучительная. Представьте себе, как только не ругались пользователи, особенно не опытные, когда пытаешься ставить один пакет, но он этого не хочет, так как сначала нужно поставить второй пакет. Ставишь второй – он требует в зависимостях третьего, третий четвертого и так далее. И каждый пакет нужно было искать вручную, часто в интернете, тогда еще очень медленном. Десятки нерешенных зависимостей заставили разработчиков задуматься: «А не написать ли нам такую программу, которая сама решает, что ставить, как и куда?». И в светлейших головах хороших людей родился первый пакетный менеджер. От человека требовалось всего лишь знать название нужного пакета, доступ к хранилищу пакетов (репозиторию) и права для самой установки. С тех пор основные принципы не изменились, но сами они значительно обросли новыми функциями, невероятной мощью и гибкостью. На сегодня самые популярные это: yum, urpmi – ставят пакеты rpm; apt – ставят в основном deb-пакеты, но иногда к ней прикручивают rpm. Сегодня я хочу немного рассказать про менеджер urpmi, который используется в любимой мной Mandriva Linux. Команда rpm предоставляет нам очень большие возможности работы с пакетами, но самая большая проблема в том, что она сама не умеет ставить зависимости. Для решения этой проблемы в недрах компании Mandriva был разработан программный комплект urpmi, который работает поверх стандартной rpm как оболочка. В состав пакета входят несколько утилит для работы с пакетами и их источниками. Ниже мы подробно рассмотрим работу каждого составляющего.

ИНСТАЛЛЯЦИЯ ПАКЕТОВ URPMI [ИМЯ_ПАКЕТА]

Для установки пакета, как я уже говорил, нужно знать его название. Для примера попробуем поставить программу для создания меню DVD-дисков `2mandvd`:

```
[root'0'@mandrivka keed]$ urpmi 2mandvd
Наступні пакунки буде встановлено для задоволення залежностей:
Пакунок Версія Випуск Арх.
(носій «Contrib media»)
2mandvd 1.3.3 4mdv2010.1 i586
dvdauthor 0.6.18 1mdv2010.1 i586
ffmpegthumbnailer 2.0.1 1mdv2010.1 i586
libffmpegthumbnailer4 2.0.1 1mdv2010.1 i586
Буде використано 4.7МБ додаткового місця на диску.
Буде отримано 2.4МБ пакунків.
Продовжити встановлення 4 пакунків? (Y/n) (Т/н)
```

Как видим, программа подробно информирует нас о том, что будет делать. Для обеспечения работы нашего пакета требуется еще три дополнительных. Если нас все устраивает, то отвечаем на запрос положительно, нажимая клавишу «y» и пакеты установятся. Можно таким же образом устанавливать одиночные rpm-пакеты, указав их расположение. Как вы уже догадались, все зависимости решаются, в отличии от обычной `rpm -Uvh [package.rpm]`. Установим, таким образом, программу для чистки ненужных эскизов – `clean`:

```
[root'0'@mandrivka keed]$ urpmi ~/clean-0.2-1.noarch.rpm'
встановлюється clean-0.2-1.noarch.rpm з /home/keed/
Підготовка... #########
############################################################
1/1: clean #########
############################################################
[root'0'@mandrivka keed]$
```

УДАЛЕНИЕ ПАКЕТОВ URPME [ИМЯ_ПАКЕТА]

Удаление также не сложно. Принцип работы такой же, как и в urpmi, только результат обратный. Для примера давайте удалим пакет `clean`, который мы установили минутой раньше:

```
[root'0'@mandrivka keed]$ urpme clean
вилучається clean-0.2-1.i586
вилучається пакунок clean-1:0.2-1.i586
[root'0'@mandrivka keed]$
```

Как бы тоже не очень сложно. Но тут иногда нужно быть осторожным, иначе по тем же зависимостям можно снести пол-системы. Для ясности попробуем удалить пакет `libgnome-desktop-2_17`, к которому много чего привязано:

```
[root'0'@linux.local keed]$ urpme libgnome-desktop-2_17
Щоб задовольнити залежності, буде вилучено 23 пакунків (170МБ):
eog-2.30.1-1mdv2010.1.i586
(через відсутність libgnome-desktop-2.so.17)
evolution-2.30.2-1mdv2010.1.i586
(через відсутність libgnome-desktop-2.so.17)
gdm-2.30.2-12mdv2010.1.i586
(через відсутність gnome-settings-daemon)
gnome-applets-2.30.0-1mdv2010.1.i586
(через відсутність libgnome-desktop-2.so.17, через відсутність gnome-python-
```

ЗАДАЧА	ПАКЕТНЫЙ МЕНЕДЖЕР			
	apt (deb) Debian, Ubuntu	zypp (rpm) openSUSE	yum (rpm) Fedora, CentOS	urpmi (rpm) Mandriva
УПРАВЛЕНИЕ ПАКЕТАМИ				
Установить пакет с репозитория	apt–get install pkg	zypper install pkg	yum install pkg	urpmi pkg
Установить ПО с файла	dpkg –i pkg	zypper install pkg	yum localinstall pkg	urpmi pkg
Обновить версию ПО	apt–get install pkg	zypper update –t package pkg	yum update pkg	urpmi pkg
УПРАВЛЕНИЕ ПАКЕТАМИ				
Удалить ненужное ПО	apt–get remove pkg	zypper remove pkg	yum erase pkg	urpme pkg
Обновить список пакетов	apt–get update aptitude update	zypper refresh	yum check–update	urpmi.update –a
Обновить систему	apt–get upgrade aptitude safe–upgrade	zypper update	yum update	urpmi –auto–select
ПОИСК В БД ПАКЕТОВ				
Поиск пакета по имени	apt–cache searchpkg	zypper search pkg	yum list pkg	urpmq pkg
Поиск по содержимому	apt–cache searchpattern	zypper search –t pattern pattern	yum search pattern	urpmq –fuzzy pkg
Поиск по файлу	apt–file search path	zypper wp file	yum provides file	urpmf file
Список установленных пакетов	dpkg –l	zypper search –is	rpm –qa	rpm –qa
РАБОТА С РЕПОЗИТОРИЯМИ				
Вывести список	cat /etc/apt/sources.list	zypper repos	yum repolist	urpmq –list–media
Добавить источник	(edit /etc/apt/sources.list)	zypper addrepo pathname	(add repo to /etc/yum.repos.d/)	urpmi. addmedianame path
Удалить источник	(edit /etc/apt/sources.list)	zypper removereponame	(remove repo from /etc/yum.repos.d/)	urpmi. removemediamedia

```
applet)
  gnome-control-center-
2.30.1-5mdv2010.1.i586
          (через
відсутність libgnome-
desktop-2.so.17,
          через
незадоволеність gnome-
settings-daemon >= 2.21.5,
          через
відсутність libgnome-
window-settings.so.1)
  gnome-panel-2.30.2-
1mdv2010.1.i586
          (через
відсутність libgnome-
desktop-2.so.17,
          че-
рез відсутність gnome-
screensaver,
          через
відсутність gnome-session)
  [ *=*=*= урезано для эко-
номии места =*=*=* ]
  Вилучити 23 пакунки?
(т/н)
```

Программа умная, она знает, что без главных библиотек среды GNOME аж 23 пакета не смогут работать и будут, просто, занимать такое ценное место на диске. Поэтому и их удалить – это все–таки мудрое решение. Но мы нажимаем «n» для отмены.

ПОИСК ПАКЕТОВ – URPMQ [ИМЯ_ПАКЕТА]

Поиск в БД пакетов происходит по таким критериям: по названию пакета (`urpmq [имя_пакета]`), по файлу в нем содержащегося (`urpmf [имя_файла]`) и в описании пакетов по ключевым словам (`urpmq –fuzzy [ключевое_слово]`). Последние два пункта требуют в наличии закэшированого файла `xml–info`, который должен находиться в папке с подключеным репозиторием. Где это, смотрите ниже.

ДОБАВЛЕНИЕ ИСТОЧНИКОВ ПАКЕТОВ – URPMI ADDMEDIA [НАЗВАНИЕ_ИСТОЧНИКА] [ПУТЬ]

Эта команда позволяет добавлять новые репозитории для установки пакетов. Добавлять можно как локальные носители, так и сетевые. Для просмотра уже добавленых можно воспользоваться этой документированной командой:

```
  [root'0'@mandrivka ~]$
urpmq –list–media
  Main media
  Contrib media
  distrib
```

У меня на данный момент подключено аж три источника – это для примера, на самом деле их больше. Хотя есть еще один, более хитрый способ получить этот список:

```
  [root'0'@mandrivka ~]$
find /var/lib/urpmi/ –type d
  /var/lib/urpmi/
  /var/lib/urpmi/Main media
  /var/lib/urpmi/distrib
  /var/lib/urpmi/Contrib
media
```

Как вы уже могли догадаться, каждый добавленный носитель предоставляет собой папку в директории `/var/lib/urpmi/`, в которой лежит список пакетов, место, где эти пакеты брать и MD5–суммы для проверки целостности. Так давайте добавим источник с названием Contrib local media из локального cdrom:

```
  [root'0'@mandrivka ~]$
urpmi.addmedia Contrib\
local\ media cdrom://i586/
media/contrib
  додавання носія «Contrib
local media»
  [root'0'@mandrivka ~]$
```

После добавления желательно обновить центральную БД пакетов командой `urpmi.update –a`. Только после этого действия пакеты с новых источников станут видимыми и доступными для установки.

УДАЛЕНИЕ ИСТОЧНИКОВ – URPMI.REMOVEMEDIA [НАЗВАНИЕ_ИСТОЧНИКА] [ПУТЬ]

Принцип действия обратно пропорциональный предыдущей команде, и синтаксис похож. Поэтому долго зацикливаться на этом не буду, просто приведу пример:

```
  [keed'0'@mandrivka ~]$
sudo urpmi.removemedia
«Contrib local media»
  вилучається носій
«Contrib local media»
  [keed'0'@mandrivka ~]$
```

И удалять можно сразу несколько источников.

RPMDRAKE

Это официальная «морда» для urpmi, написанная на связке perl+gtk. Довольно удобная оболочка, но, по–моему, еще не так функциональна, как, например, synaptic. Пакеты удобно собраны в группы и отображаются в левой колонке. Правая же колонка горизонтально поделена на две части. В верхней выбираем интересующий нас пакет, а снизу читаем информацию про него. Запутаться трудно, но возможно – я один раз долго не мог понять, почему rpmdrake никак не находил mplayer. А дело было в том, что сразу после установки системы он настроен на отображение пакетов с GUI. Просто в выпадающем списке над левой колонкой выбираем «Все» и будет вам счастье

В завершение приведу русифицированную табличку, которую я бессовестно стащил с сайта http://distrowatch.com/. В ней можно найти основную информацию о командах и их применении. А также используйте ее как словарь–переводчик с других пакетных менеджеров.

P.S. Учите консольные команды. У меня один раз в самый ответственный момент rpmdrake просто рухнул и до сих пор не запускается. И где гарантия того, что это с вами не случится? Если изложенной мной информации вам мало, то просто запустите любую команду без параметров и получите хорошее, толковое руководство. Благо оно русифицировано и легко для понимания. И вы поймёте, насколько мощным инструментом наградили разработчики мандриву!

Андрей 'keedhost' Кондратьев
http://keedhost.blogspot.com

UA Linux

open source ■ open future

* разработка комплекта дополнений Ubuntu Applications Pack

* разработка, внедрение и сопровождение эффективных IT-решений на базе открытого ПО

* IT-аутсорсинг

* поддержка и консультирование пользователей

* внедрение систем виртуализации

* тестирование оборудования, а также доработка под него программного обеспечения

Внимание!!!

С приходом лета, приходят хорошие новости!

Мы объявляем подписку на печатный журнал UserAndLINUX!

Следите за анонсами на странице журнала http://ualinux.com/index.php/journal