Фактический материал

В любой даже самой отточенной системе всегда есть резерв для совершенствования. Этот раздел поможет вам сделать работу в Linux удобной и продуктивной, возможно, здесь вы найдете решение какой-либо программной или аппаратной проблемы. Начнем с советов по работе с графическими средами GNOME и KDE.

Обе среды довольно удобны, но, скорее всего, вам наверняка чего-то не хватает. Возможно, это какое-то действие в пункте меню, которое вы часто выполняете вручную и хотели бы автоматизировать этот процесс, или упаковка файла каким-то специфическим архиватором, или изменение прав доступа к нему.

KDE

Создание собственных действий в сервисном меню

Давайте создадим дополнительную команду, которая делает файл исполнимым. (Как говорится, был обычный файл, а стал программой.) Это действие может оказаться очень полезным, если вы часто пишете сценарии bash. Ведь сценарий bash — это обычный текстовый файл, созданный в текстовом редакторе. Чтобы он запускался, нужно сделать его исполнимым. Данный совет посвящается любителям все делать мышкой — тем, кому лень вводить команду вида «chmod +x имя_файла».

Итак, в любимом текстовом редакторе создайте файл следующего содержания:

```
[Desktop Entry]
ServiceTypes=all/allfiles
ServiceType=application/x-shellscript
Actions=MakeExe
[Desktop Action MakeExe]
Name=Make executable
Name[ru]=Сделать файл исполнимым
Exec=chmod +x %f
Icon=kfm
```

Рассмотрим первую секцию. Первая опция задает тип файлов, для которых можно выполнить указанное действие. В данном случае оно доступно для всех файлов (allfiles). Если нужно выполнить какое-то действие для каталога, то параметр ServiceTypes должен принять значение inode/directory:

```
ServiceTypes=inode/directory
```

Вообще, в качестве значения данного параметра можно указать любой MIME-тип, например:

```
ServiceTypes=audio/x-mp3
```

Если требуется выполнить какое-то действие для всех типов файлов кроме каких-то определенных, используется параметр ExcludeServiceTypes. Например, если вы определяете действие архивирования, то должны указать сервисные типы — все файлы, но исключить архивы:

```
ServiceTypes=all/allfiles
ExcludeServiceTypes=application/
x-zip,kdedevice/*
```

Параметр Actions определяет действия, описанные в файле. В данном случае описано только одно из них — MakeExe, определенное в секции [Desktop Action MakeExe]. Параметр Name — это надпись, которую вы найдете в сервисном меню KDE. Это общая надпись, которую увидит пользователь, ее желательно писать на английском языке. Параметр Name[ru] — это тоже надпись, описывающая действие, но на русском языке. Пользователь увидит ее, если у него KDE на русском языке.

Параметр Exec — это команда, которая будет выполнена, а %f — параметр, определяющий имя файла, по которому вы щелкнули правой кнопкой; нужное имя будет подставлено вместо %f.

Созданный файл сохраните под именем make_exe.desktop.

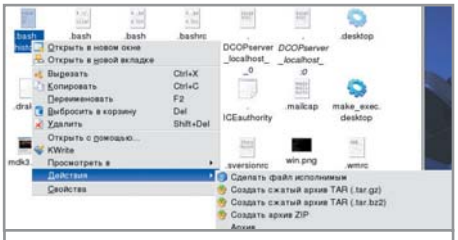
Как видите, в этом нет ничего сложного. Осталось только сохранить файл в нужном каталоге — servicemenus. Он находится в каталоге /usr/share/apps/conqueror/:

```
$ sudo cp make_exe_desktop
/usr/share/apps/conqueror/servicemenus/
```

Для записи в этот каталог нужны права суперпользователя, поэтому, чтобы выполнить команду cp, понадобится команда sudo, которая и обеспечит нужные полномочия (вам нужно будет ввести пароль пользователя root). Если же вы не администратор системы и хотите добавить команду в меню только локального пользователя, скопируйте файл make_exe_desktop в каталог ~/.kde/share/apps/conqueror/servicemenus/:

```
$ cp make_exe_desktop
~/.kde/share/apps/conqueror/service-
menus/
```

Результат вашей работы появится в параметре Actions сервисного меню KDE.



Создание сервисного меню

GNOME

Меню «Сценарии»

Меню «Сценарии» аналогично меню «Действия» в KDE. Идентичен и принцип работы. Мы определяем команды, которые будут выполнены при выборе того или иного пункта меню. Только если в случае с KDE файл меню имел определенный формат, то для GNOME файл

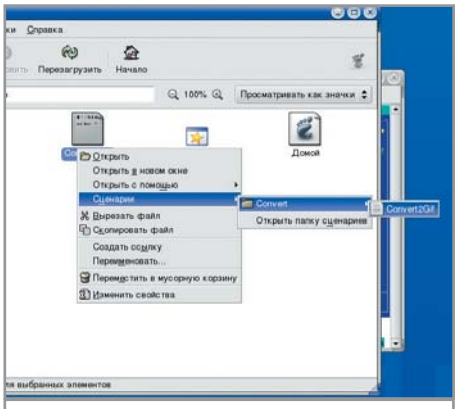
меню — это обычный bash-сценарий. Хорошо это или плохо, зависит от степени вашего знакомства с bash.

Если вы знаете его хорошо, то сможете создавать очень сложные сценарии, позволяющие автоматизировать огромное количество рутинной работы. Действия вашего сценария не будут ограничены форматом файла, как в KDE. Ведь, по сути, в KDE для выполнения действия мы можем определить только простые команды. Если нужно использовать то же разветвление (if ... then), все равно придется писать сценарий на bash, а из меню KDE вызывать не какую-то программу, а именно созданный вами сценарий.

Если же вы не очень хорошо знакомы с bash, рекомендуем его выучить, а пока вы будете это делать, лучше создавать дополнительные пункты меню в KDE.

Идея создания пункта меню достаточно проста. Вы пишете bash-сценарий и помещаете его в каталог ~/gnome2/nautilus-scripts/. В нем же можно создавать подкаталоги — они будут отображаться как дополнительные меню. На скриншоте видно, что в каталоге nautilus-scripts был создан подкаталог Convert, а в нем — файл Convert2Gif. Думаем, с этим все ясно. Теперь рассмотрим пример самого сценария:

```
#!/bin/bash
convertprg=`which convert`
while [ $# -gt 0 ]; do
        picture=$1
        filetype=`file $picture | cut -d ' '
-f 3`
        if [ $filetype = "image" ]
        then
                newfile=`echo "$pic-
ture" | cut -d . -f 1`
                $convertprg "$pic-
ture" "$newfile".gif
        fi
        shift
done
```



Меню «Сценарии» в GNOME

Данный сценарий выполняет конвертирование изображения в формат GIF. В качестве программы-конвертера используется convert.

Файл, по которому пользователь щелкнул правой кнопкой мыши, передается нашему сценарию как первый параметр:

```
picture=$1
```

При написании сценариев действий вы можете использовать следующие переменные окружения, которые устанавливаются средой GNOME.

- NAUTILUS_SCRIPT_SELECTED_FILE_PATHS — имена всех выбранных файлов (в случае если вы выбрали группу файлов, а затем — созданное вами действие) будут разделены символом новой строки (\n). Данная переменная работает только с локальными файлами. Если вы выберете файлы, например, в каталоге FTP-сервера, то эта переменная устанавливаться не будет.
- NAUTILUS_SCRIPT_SELECTED_URIS — эта переменная как раз предназначена для работы с удаленными файлами и содержит URI файлов, разделенных символом новой строки.
- NAUTILUS_SCRIPT_WINDOW_GEOMETRY — данная переменная содержит геометрию (позицию и размер) текущего окна Nautilus.

KDE

Программа kstart

Программа kstart позволяет управлять запуском приложения в графической среде KDE. Рассмотрим несколько примеров ее использования (подробное описание этой программы вы найдете в справочном руководстве).

Для того чтобы запустить приложение Konqueror на втором виртуальном рабочем столе, достаточно выполнить следующую команду:

```
kstart --desktop 2 konqueror
```

Программу также можно запустить сразу на всех рабочих столах, для этого используется опция alldesktops:

```
kstart --alldesktops konqueror
```

Чтобы запустить все тот же Konqueror и развернуть его во весь экран, используется опция fullscreen:

```
kstart --fullscreen konqueror
```

И, наконец, для запуска на всех рабочих столах приложения noatun в режиме OnTop (всегда поверх всех окон) используется такая команда:

```
kstart --alldesktops --ontop noatun
```

GNOME

Автоматическая смена обоев

Наверное, многие из нас видели программу Webshots Desktop. Жаль, что она работает только под Windows. Хотелось бы, чтобы у нее была Linux-версия. Но пока ее нет, мы попробуем написать ее аналог.

Для тех, что не знает, что это за программа, вкратце расскажем, как она работает. Существует каталог с обоями рабочего стола — /home/den/Wallpapers/. Каждый раз при запуске GNOME менять обои вручную неудобно. Нужно автоматизировать эту рутинную процедуру, и в этом нам поможет следующий сценарий:

```
#!/bin/bash
export DIR='/home/den/Wallpapers/'
export NUMBER=$RANDOM
export TOTAL=0
for f in `ls $DIR`
do
        let "TOTAL += 1"
done
let "NUMBER %= TOTAL"
export CURRENT=0
for f in `ls $DIR`
do
        if [ $CURRENT = $NUMBER
]
        then
                /usr/bin/gconftool-2 -
t string -s /desktop/gnome/back-
ground/picture_filename $DIR/$f
                break
        fi
        let "CURRENT += 1"
done
```

Особо вникать в этот сценарий не нужно. Достаточно понять, что он с помощью команды gconftool-2 уста-

Автозапуск GNOME

навливает файл обоев, который выбирается случайным образом в первой части сценария из каталога, заданного переменной $DIR.

Сохраните этот файл под именем change_wp и сделайте его исполнимым:

$ chmod +x change_wp

Для изменения обоев вам достаточно выполнить следующую команду (обои будут изменены мгновенно):

./change_wp

А теперь, собственно, займемся автоматизацией, ведь вы же не будете вводить эту команду каждый раз при запуске GNOME. Тут можно поступить двумя способами: или поместить вызов этого сценария в таблицу crontab (должен быть запущен демон crond), или добавить команду вызова этого сценария в автозапуск непосредственно GNOME. Crond предлагает больше возможностей. Например, вы можете установить смену обоев каждый час. А в другом случае обои будут меняться только при перезагрузке, то есть всякий раз, когда вы регистрируетесь в GNOME.

Графика

Снимок экрана из командной строки

Снимок экрана (или какого-то окна) можно сделать многими программами: например, очень хороша KSnapshot, входящая в состав KDE. Также снимок экрана можно сделать и с помощью GIMP. Но иногда эти программы не доступны — например, у вас вообще не установлен KDE, а использовать GIMP для снимка экрана/окна, мягко говоря, нерационально.

Сделать снимок экрана можно очень легко и быстро с помощью утилиты import, которая является частью пакета ImageMagic (этот пакет должен быть у вас установлен). Рассмотрим следующую команду:

$ sleep 5; import -window root screen.png

Ее можно ввести или в X-терминале, или в окне запуска программы KDE (для вызова окна нажмите сочетание клавиш «Alt+F2»). Данная строка на самом деле состоит из двух команд — sleep и import. Команда sleep генерирует необходимую задержку в секундах. Думаем, пяти секунд хватит, чтобы вы смогли привести экран в надлежащий вид: запустить или активировать нужное окно, выбрать подходящий пункт меню и т. д.

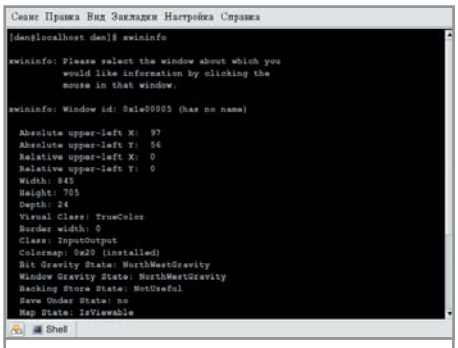
Команда import будет выполнена сразу после обработки sleep, то есть спустя пять секунд. Она делает снимок корневого окна, то есть всего экрана, и записывает его в файл screen.png. Формат PNG наиболее оптимален для создания снимка экрана. Если вам нужен другой формат, например JPG, просто измените расширение результирующего файла:

$ sleep 5; import -window root screen.jpg

Но не всегда нужен снимок всего экрана, в большинстве случаев необходимо лишь какое-то конкретное окно. Запускать потом приложение GIMP и вырезать нужное окно из общей «фотографии» не очень удобно — проще с самого начала сделать снимок определенного окна.

Первый способ — указать координаты окна, например:

$ sleep 5; import -crop 400x300 screen.png



Программа XWinInfo. Этот снимок сделан с помощью команды import

Второй способ — указать геометрию окна:

$ sleep 5; import -geometry геометрия screen.png

Вычислить координаты и геометрию позволяет команда xwininfo.

После запуска программы XWinInfo указатель мыши поменяет свой вид со стрелки на крестик. Вы должны выбрать окно, о котором хотите получить информацию, после чего программа вам их предоставит. Геометрия окна выводится почти в самом конце, поэтому на рисунке она не видна.

И, наконец, рассмотрим третий, самый простой способ сделать снимок определенного окна:

$ import window.png

Команда sleep не понадобится, поскольку программа import предоставит вам возможность выбрать нужное окно: указатель мыши изменится на крестик, как и в случае с программой import. Щелкните по нужному окну, и буквально через полсекунды в файл window.png будет записан его образ.

Программа import довольно гибкая, рекомендуем прочитать ее man — вы найдете там много интересного.

А что если вам нужно сделать снимок текстовой программы, а не графического окна? Проще всего запустить ее в X-терминале, а затем снимать окно X-терминала, как показано выше. Если обрамление окна X-терминала вам не нужно, его можно вырезать с помощью GIMP.

Администрирование

Клонирование системы

Клонирование — это очень полезный процесс. Предположим, вы только что установили Linux на один компьютер в интернет-зале. Потом посмотрели вокруг и осознали, что вам предстоит повторить этот процесс минимум еще на пяти-семи машинах. Вот тут как раз и целесообразно использовать клонирование — создание точной (побитной) копии исходного носителя. В этом случае в качестве носителя будет выступать корневая файловая система Linux. Клонированная копия называется образом. Мы попробуем сделать точную копию, а потом развернуть ее

на других компьютерах в зале. Надеемся, вы понимаете, что такая операция пройдет корректно, если у всех компьютеров будет одинаковая конфигурация. В случае с интернет-залом обычно так оно и есть. Конечно, можно развернуть образ и на компьютере, конфигурация которого отличается от исходного, но потом он может потребовать дополнительной настройки — вплоть до перекомпиляции ядра. А на все это уйдет намного больше времени, чем на установку системы с нуля, которая на современных компьютерах занимает не более получаса.

Не подумайте, что клонирование полезно только тогда, когда у вас много одинаковых компьютеров. Его целесообразно использовать, даже если у вас всего один компьютер. Например, вы можете сделать образ домашней системы. Если что-то вдруг слетит, вы легко и быстро восстановите исходное состояние системы простым развертыванием образа. Или если у вас что-то случилось с сервером, можно очень быстро «понять» его — ведь на развертывание образа нужно гораздо меньше времени, чем на установку и настройку системы. Время простоя в этом случае будет минимальным.

Итак, приступим непосредственно к клонированию. Перезагрузитесь в однопользовательском режиме. Для этого нужно использовать параметр single ядра Linux. Введите команду mount, чтобы узнать, какой раздел содержит корневую файловую систему:

```
/dev/hda1 on / type ext3 (rw,noatime)
none on /proc type proc (rw)
none on /sys type sysfs (rw)
```

Корневая файловая система расположена в разделе /dev/hda1. Создадим каталог /mnt/image — он нам скоро понадобится:

```
# mkdir /mnt/image
```

К этому каталогу нужно подмонтировать носитель, на который будет записываться образ. Понятно, что в директорию /dev/hda1 записать образ вы не сможете, поскольку образ именно этого раздела и собираетесь сейчас делать. Носителем может быть другой жесткий диск. Мы же будем использовать внешний USB-винчестер. Для его подключения нужно загрузить модуль usb_storage. Практически во всех новых дистрибутивах он уже откомпилирован.

```
# modprobe usb_storage
# mount /dev/sda1 /mnt/image
```

Первая команда загружает модуль (если он еще не загружен), а вторая монтирует устройство /dev/sda1 (это и есть внешний винчестер) к каталогу /mnt/image. Все, что осталось сделать, — это смонтировать корневую файловую систему в режиме «Только чтение» и создать образ:

```
# sync
# mount -o remount,ro /
```

Создаем образ /mnt/image/image.bin раздела /dev/hda1:

```
# dd if=/dev/hda1
of=/mnt/image/image.bin
```

Подробно описывать утилиту dd мы не станем — это очень хорошо сделано в справочной системе. Скажем только, что кроме dd есть еще и утилита dd_rescue, которая при клонировании пропускает плохие секторы.

Теперь рассмотрим, каким образом можно восстановить систему. Для этого вам понадобится загрузочный диск Linux. Подойдет первый диск любого дистрибутива (при условии, что он загрузочный) — Mandrake, Red Hat; можно также использовать Knoppix CD или Gentoo LiveCD.

Если вы используете не LiveCD, а простой загрузочный диск, для перехода в консоль нажмите «Ctrl+Alt+F2». Теперь подмонтируем внешний винчестер:

```
# mkdir /image
# modprobe usb_storage
# mount /dev/sda1 /image
```

Выше приведенные команды (при условии, что второй жесткий диск подключен как Primary Slave) будут выглядеть так:

```
# mkdir /image
# mount /dev/hdb1 /image
```

Теперь вам нужно создать разделы на новом винчестере. Это можно сделать с помощью утилиты fdisk или программы установки — как вам удобнее. Если же вы восстанавливаете систему после сбоя, ничего создавать не нужно — все уже есть. Не забудьте только создать раздел подкачки (тип раздела /dev/hda2 должен быть Linux swap):

```
# mkswap /dev/hda2
```

Далее развернем образ (обратите внимание на параметры if и of программы dd — теперь их аргументы поменялись местами):

```
# dd if=/foo/image.bin of=/dev/hda1
```

Сейчас нам нужно изменить корневую файловую систему, чтобы попасть «внутрь» развернутого образа:

```
# mkdir /install
# mount /dev/hda1 /install
# chroot /install /bin/bash
```

Первая команда создает каталог install, вторая монтирует «подопытный» раздел к этому каталогу, а третья изменяет корневую файловую систему. Теперь корнем стал каталог install, а в качестве командной оболочки используется директория /bin/bash. Все, что нам осталось сделать, — это перезаписать загрузчик. Если вы используете LILO, введите следующую команду:

```
# lilo
```

А если GRUB — то такую:

```
# grub-install /dev/hda
```

Теперь перезагрузите компьютер (команда reboot) — ваша система успешно клонирована/восстановлена!

Администрирование

Простое резервное копирование по сети

Предположим, что вы — сетевой администратор. В ваши обязанности входит также и резервное копирование пользовательских данных, то есть каталогов /home с каждого компьютера сети. Подходить к каждому компьютеру — не хочется, поэтому лучше автоматизировать процесс.

Скопировать каталог /home/den, расположенный на компьютере denis, можно с помощью этой команды:

```
scp -r backup-den denis:/home/den
```

Параметр -r означает, что будут копироваться также и подкаталоги удаленного каталога. Часть команды backup-den — это имя каталога, куда будет записана резервная копия; Denis — имя компьютера (можно использовать IP-адрес, например, 192.168.1.5); /home/den — это удаленный каталог.

Что же за команда scp? Она расшифровывается secure copy. Для ее работы нужно, чтобы на компьютере Denis был запущен демон sshd. В свою очередь, sshd спросит вас о пароле и имени пользователя — и вы должны ответить на этот вопрос.

Тюнинг

Повышаем производительность системы

Для повышения производительности системы вам не понадобится перекомпилировать ядро. Данный совет рассчитан только на версию ядра 2.6.

Оптимизацию производительности начнем с установки оптимальных для нас параметров виртуальной памяти.

В псевдофайле /proc/sys/vm/swappiness содержится коэффициент подкачки. Что это такое? Предположим, что вы работаете с несколькими (или даже одним) довольно громоздкими приложениями и нечасто переключаетесь между ними. Возможно, вы дизайнер — с утра вы запускаете GIMP и не выходите из него до самого вечера. Также утром вы запускаете xmms — как работать без музыки? Изредка, скажем раз пять-семь за день, вам нужно переключаться между GIMP и xmms, чтобы выбрать новый каталог или новую песню.

Если вы установите большое значение коэффициента подкачки, например 90 или даже 100 (максимальное), то переключение между этими приложениями будет происходить довольно медленно, но зато производительность основного приложения (то есть GIMP) заметно улучшится.

Если вы целый день работаете с небольшими программками и часто переключаетесь между ними, вам лучше установить коэффициент в районе 20 или 30.

Поэкспериментируйте с различными параметрами и выберите оптимальный для себя. Вывести значение файла /proc/sys/vm/swappiness можно с помощью команды:

```
# cat /proc/sys/vm/swappiness
```

Значение по умолчанию — 70. Возможно, вам больше подойдет именно такое.

Установить значение (в данном случае 20) можно с помощью команды:

```
# echo "20" > /proc/sys/vm/swappiness
```

Все действия должны осуществляться только с правами суперпользователя (root).

Теперь займемся повышением производительности сети. Если у вас обычный домашний компьютер, подключенный к Интернету по *DSL или Ethernet, имеет смысл попробовать выключить некоторые параметры, это можно сделать следующими командами:

```
# echo "0" > /proc/sys/net/ipv4/tcp_sack
# echo "0" >
/proc/sys/net/ipv4/tcp_timestamps
```

Если результат вас не устроит, значение параметров можно вернуть («1»).

А теперь приступим к самому важному. У вас есть отличный шанс существенно повысить производительность системы.

Каждой программе, работающей под Linux, время от времени необходим доступ к диску. Ядро Linux определяет, когда именно программа получит доступ к диску. Часть ядра, отвечающая за это, называется планировщиком ввода/вывода. Есть четыре алгоритма его работы.

- Режим по умолчанию (noop) — вряд ли он подойдет для обычного пользователя, несмотря на то, что используется по умолчанию. Рассматривать его мы не будем.
- Упреждающее планирование (Anti-cipatory Scheduling) — при чтении программой данных с диска ядро пытается предугадать, какие данные программа будет читать при следующей операции. Кроме всего прочего эффективность этого алгоритма сильно зависит от логики программы. Параметр ядра — elevator = as.
- «Справедливая» очередь (Complete Fairness Queuing) — равные права для всех программ. Ядро равномерно планирует операции ввода/вывода для каждого приложения, здесь нет каких-либо программ, которые могут монополизировать доступ к диску. Если несколько программ одновременно запросят доступ к диску, все они получат ответ. Данный метод во многих случаях позволяет повысить производительность системы, а в некоторых, наоборот, снижает общую производительность — все зависит от конкретной ситуации. Параметр ядра — elevator = cfq.
- Deadline-планирование, или планирование крайних сроков (Deadline Queuing) — все приложения, запросившие доступ к диску, ставятся в очередь. Из очереди извлекается одна программа, которая и получает практически монопольный доступ к диску. Пока эта программа работает, все остальные ожидают очереди. По истечении определенного времени планировщик переводит эту программу в состояние ожидания и переключается на другую — следующую по очереди. Теперь вторая программа получает безраздельный доступ к диску. Потом третья, четвертая и т. д. Данный метод оптимален для сервера баз данных, но не для пользовательского компьютера. Параметр ядра — elevator = deadline.

У каждого алгоритма есть преимущества и недостатки. Но только два алгоритма подходят для обычного домашнего компьютера (рабочей станции) — второй и третий. В Интернете вы можете найти данные о том, что для настольного компьютера более подходит второй алгоритм. Решать вам — все познается в сравнении, тем более, что вы ничем не рискуете. Для применения нового алгоритма достаточно указать определенный параметр ядра и немного поработать с системой — если не понравится, всегда можно попробовать другой вариант. Вам не нужно перекомпилировать ядро — необходимо просто пару раз перезагрузить компьютер. Если вы подберете нужный алгоритм, тогда вам понадобится перезаписать загрузчик (в случае с LILO).

Для выбора нужного режима перезагрузите компьютер и при запуске ядра Linux передайте ему один из указанных выше параметров. Например, для выбора упреждающего планирования нужно передать ядру параметр elevator = as:

```
linux elevator=as
```

Предположим, что вам больше всего понравился именно этот параметр. Теперь его нужно «утвердить» в файле конфигурации загрузчика. Если вы используете LILO, откройте файл /etc/lilo.conf и найдите в нем строку:

```
append = "<параметры ядра>"
```

Добавьте в эту строку параметр linux elevator = as:

```
append = "elevator = as (и другие
параметры)"
```

Например, файл может выглядеть так:

```
image = /boot/vmlinuz-2.6.9
        label = Linux
        root = /dev/hda1
        append = "elevator = as video =
vesafb:ywrap,mtrr,1024x768-16@75"
```

Сохраните файл и перезапишите загрузчик командой lilo.

Если у вас загрузчик GRUB, перезапись не обязательна. Нужно только отредактировать файл /boot/grub/grub.conf:

```
title My Default Linux
root (hd1,0)
kernel /boot/vmlinuz-2.6.9 ro root =
/dev/hda1 elevator = as
...
```