# Создание легкой системы на базе Slackware

## I - Введение

Slackware пользуется заслуженной популярностью как классический linux дистрибутив, и поговорка "кто знает Red Hat тот знает только Red Hat, кто знает Slackware тот знает linux" несмотря на явный снобизм поклонников "бога Патрега" все же имеет под собой основания. Одним из преимуществ Slackware является возможность простого создания на ее основе практически любой системы, в том числе быстрой и легкой десктопной, о чем далее и пойдет речь. Есть дистрибутивы, клоны Slackware, созданные именно с этой целью, типа Absolute, но все же лучше создавать систему под себя, с максимальным учетом именно своих потребностей, и Slackware пожалуй как никакой другой дистрибутив подходит именно для этой цели.

Легкость и быстрота системы определяется выбором WM (DM) , набором программ и оптимизацией программ и системы в целом. Первое исключает KDE, Gnome, даже новые версии XFCE, остается разве что LXDE, но набор программ в нем совершенно не устраивает. Оптимизация наиболее часто используемых программ и нескольких базовых системных пакетов осуществляется их сборкой из сорцов компилятором, оптимизированным именно под Ваш комп, причем каждая программа конфигурируется исходя из Ваших потребностей к ее возможностям. Оптимизация системы в целом осуществляется ее настройкой согласно специфическим требованиям к десктопу.

Такой подход был выбран по банальной причине, возиться с gentoo нет никакого желания, комп все таки создан для того чтобы им пользоваться, а не для компиляции программ, в тоже время у каждого есть минимальный набор из небольшого количества наиболее часто используемых программ, на которые стоит потратить некоторое, не такое уж большое, время, чтобы довести их до ума. Кроме того, такой подход позволяет иметь самые свежие версии наиболее часто используемых программ.

Исходя из вышесказанного были выбраны IceWM, openbox и Enlightenment в его обеих версиях и набор наиболее часто используемых программ. Такой выбор WM объясняется довольно просто, это наиболее распространенные WM (Enlightenment вообще то DE, но его программы лучше не трогать, они малофункциональны и глюкавы), fvwm велик, могуч но и страшен, копаться в его настройках занятие не для слабонервных, остальные весьма мало распространены и доводить их до ума приходится самому с нуля, всякие тайловые изначально не рассматривались. IceWM быстр, прекрасно и легко настраиваем под любой вкус, имеет отлично документированные конфиги, но хотя его внешний вид и можно настроить самыми красивыми темами, макет всех его тем в принципе несколько схож. Openbox сейчас, пожалуй, самый массовый WM. Enlightenment E16 еще более быстр чем IceWM, имеет прекрасную графическую конфигурялку, но его внешний вид, в отличии от IceWM, можно сделать практически любым, со множеством уже имеющихся великолепных тем. Enlightenment E17 еще более красив, и практически так же быстр как IceWM. Enlightenment что называется надо "проникнуться" но потом отказаться от него очень трудно, все остальные кажутся уже какой то бледной поделкой. Почему несколько WM ответ простой - один может и надоесть.

Причем ниже сказанное применимо к любому набору Ваших любимых программ, любимому WM(DM), хоть к KDE из самой Slackware, и потребностям к их возможностям, поскольку описанные далее принципы работы, установка, настройка и оптимизация самой Slackware, методы построения такой системы применимы ко всем из них. Не зависит сказанное в основном и от версии Slackware, по крайней мере пока кардинально на поменяется логика его создания, что вряд ли вообще когда либо произойдет. От версии Slackware зависит главным образом линки на программы в соответствующих разделах сайтов, поэтому такие линки приводятся без привязки к версии слаки, также от версии может зависеть русификация. Далее написано пошагово, так что если пропустили какой то пункт и работает не так как хочется, смотрим пропущенные пункты

## II — Установка

- если есть проблемы с настройкой сети лучше сразу скачать нужный драйвер для видеокарты (Nvidia с www.nvidia.com/object/unix.html или www.nvidia.ru/Download/Find.aspx?lang=ru) и font terminus repository.slacky.eu/slackware-*.*/system/terminus-font/4.30/terminus-font-4.30-noarch-1bj.txz - готовый пакет

- загружаем образ -ftp.yandex.ru/slackware/slackware-*.*-iso/slackware-*.*-install-dvd.iso (ISO с Яндекса), где «*» это номер версии Slackware.

- записываем на dvd

- подготавливаем разделы для установки (минимум 19Гб под /) при помощи какого нибудь liveCD где есть gparted. Выбор файловой системы сейчас фактически ограничен ext3 и ext4. Xfs и jfs имеют весьма интересные глюки и для корня их лучше не ставить, для разделов со всякой мультимедиа и большими файлами они вполне пригодны, но не для корня. Reiser3 не имеет особых преимуществ перед ext4 и даже ext3, зато имеет массу неприятных особенностей, reiser4 вообще не поддерживается ядром, да и перспективы этой ФС весьма туманны, по крайней мере пока Ганс Рейзер припухает на нарах. Btrfs на десктопе не имеет преимуществ перед ext, зато и по сей день находится в стадии разработки, со всеми приятными неожиданностями для такой стадии. Поэтому выбор по сути между ext3 и ext4. Обе стабильны, ext4 имеет свои преимущества, но требует больше ресурсов чем ext3. Если комп достаточно мощный то выбор за ext4, если слака ставится на всякую древность типа третьих пней или самых первых четвертых, то тогда лучше ext3. Крайне рекомендуется создать еще один линукс раздел, кроме корня и свапа, который будет использоваться для компиляции программ и хранения созданных пакетов, на нем же будет создаваться архив раздела с самой слакой. Размер этого дополнительного раздела от 19 Гб.

- загружаемся с DVD (не загружается установите в bios первым загрузку с CD)

- отвечаем на вопрос о раскладке (все по умолчанию - Enter, русификация потом)

- жмем Enter на выборе ядра (ядро по умолчанию лучше оставить)

- root (здесь можно и разделы создать, но cfdisk явно не лучшее средство для этого)

- setup

- выбираем пункт "ADDSWAP option" в меню, выбираем раздел для свапа

- выбираем раздел(ы) для установки

- добавляем в fstab разделы винды

- SOURCE - пусть сам ищет DVD (CD), как правило находит

- SELECT - выбираем что ставить
Сразу выкидываются E-GNU Emacs, KDE, KDEI-Language support KDE, T-TeX, TCL, Y-Classic text-based BSD games

- expert mode

Выкидываем пакеты

- cups (если принтера нет)
- mysql

- clisp (если не фанатик лиспа), gcc-fortran,gcc-gnat, (если не хотите программировать на ада и фортране),туда же и ruby

- pycups (если не нужна печать), system-configuration-ptinter (если нет принтера) httpd (апач как то не нужен), php в компанию к апачу и мускулю, proftpd, samba (если не нужна связь с виндовыми компами), sendmail, vsftpd

- ставится все

- mplayer (если выкидывается samba или если будет собираться из сорцов, как сами авторы рекомендуют), audacious (если есть претензии к качеству звука и функционалу, меня они есть.), sane (если сканер не нужен), seamonkey (если нет ностальгии по старой мозилле), thunar-volman, весь xfce (который теперь по прожорливости мало отличается от gnome), xsane

- пьем кофе - коньяк - пиво, курим

настройка (не по порядку и кроме очевидных, типа выбора часового пояса)

- utf-8 в консоли - да

- LILO - auto mode (добавляем раздел с установленной Slackware и, если есть, раздел с Windows, ставим lilo в MBR)

- ставим русский шрифт Cyr_a8x16

- настройка сети, стандартная (ADSL модем роутером - комп 198.162.1.2, маска 255.255.255.0, gateway 192.168.1.1, DNS своего провайдера)

- убираем ненужные сервисы, ssh, bind, pcmcia (если не нужно) итп.

- задаем пароль для root

- exit

- ctrl-alt-del

- логинимся root

- startx (да-да, это некошерно , но так проще :-). Запускается fluxbox. Если не запустился, набираем xwmconfig и выбираем в нем fluxbox.

- ставим нормальные шрифты для консоли, выполняем в папке со скачанным пакетом в терминале (xterm, rxvt) команду. Не забываем что для любых файловых операций есть mc.

```
installpkg terminus-font-4.30-noarch-1tm.txz
```

- копируем /etc/X11/app-defaults/XTerm в /root/XTerm
и добавляем в него

```
XTerm*scrollBar: True
XTerm*font: -xos4-terminus-bold-r-normal--22-220-72-72-c-110-iso10646-1
XTerm*geometry: 119x39+0+1
```

- Если при установке не настроили сеть, то сейчас самое время ее настроить, она понадобится при сборке своих пакетов

```
netconfig, pppoe, etc
```

– Если сеть настроена можно сразу обновить слаку до последних актуальных версий пакетов.

Делаем:

- генерируем новый список зеркал для slackpkg

```
slackpkg new-config
```

- В /etc/slackpkg/mirrors раскомментируем **одно** зеркало (не подошло выбираем другое)

- Обновляем базу пакетов

```
slackpkg update
```

- Обновляем сами пакеты

```
slackpkg upgrade-all
```

## III - Русификация

- **Установка русской локали UTF-8**

Редактируем файл /etc/profile.d/lang.(c)sh.

```
#!/bin/sh
# en_US is the Slackware default locale:
#export LANG=en_US
# There is also support for UTF-8 locales, but be aware that
# some programs are not yet able to handle UTF-8 and will fail
# to run properly. In those cases, you can set LANG=C before
# starting them. Still, I'd avoid UTF unless you actually need it.
#export LANG=en_US.UTF-8

export LANG=ru_RU.UTF-8

# One side effect of the newer locales is that the sort order
# is no longer according to ASCII values, so the sort order will
# change in many places. Since this isn't usually expected and
# can break scripts, we'll stick with traditional ASCII sorting.
# If you'd prefer the sort algorithm that goes with your $LANG
# setting, comment this out.

export LC_COLLATE=C

# End of /etc/profile.d/lang.sh
```

- **Русская раскладка клавиатуры**

Создаем файл /etc/rc.d/rc.keymap и делаем исполняемым.

Здесь и далее помним что " \" в bash это просто перевод одной и той же команды на новую строку, если она не помещается на экране, а так это одна команда в одной строке. Это не касается xml-вещей и текстовых файлов настроек, там " \" нет, но в них по смыслу понятно, где перенос на новую строку из-за недостатка места, а где новая команда

```
#!/bin/sh
# Load the keyboard map. More maps are in
#/usr/share/kbd/keymaps.
if [ -x /usr/bin/loadkeys ]; then
  /usr/bin/loadkeys \
  /usr/share/kbd/keymaps/i386/qwerty/ruwin_ct_sh-UTF-8.map.gz
fi
```

Все файлы должны быть исполняемые.

**- Переключение раскладок по Ctrl-Shift**

```
cp /usr/share/hal/fdi/policy/10osvendor/10-keymap.fdi \
/etc/hal/fdi/policy/10-keymap.fdi
```

Редактируем файл /etc/hal/fdi/policy/10-keymap.fdi.

```
<?xml version="1.0" encoding="ISO-8859-1"?> <!-- -*- SGML -*- -->
 <deviceinfo version="0.2">
  <device>
   <match key="info.capabilities" contains="input.keymap">
   <append key="info.callouts.add" type="strlist">hal-setup-keymap</append>
  </match>
 <match key="info.capabilities" contains="input.keys">
   <merge key="input.xkb.rules" type="string">base</merge>
   <merge key="input.xkb.model" type="string">evdev</merge>
   <merge key="input.xkb.layout" type="string">us,ru</merge>
   <merge key="input.xkb.variant" type="string">,winkeys</merge>
   <merge key="input.xkb.options"
 type="string">terminate:ctrl_alt_bksp,grp:ctrl_shift_toggle,grp_led:scroll</
merge>
   </match>
 </device>
</deviceinfo>
```

В slackware версий выше 13.1 создаем файл /etc/X11/xorg.conf.d/90-keyboard-layout.conf и пишем в него (переключение раскладок по Ctrl-Shift).

```
Section "InputClass"writeback
    Identifier "keyboard-all"
    MatchIsKeyboard "on"
    Driver "evdev"
    Option "XkbLayout" "us,ru"
    Option "XkbVariant" ",winkeys"
    Option "XkbOptions" "terminate:ctrl_alt_bksp,grp:ctrl_shift_toggle,grp_led:scroll"
EndSection
```

Если хочется переключения через CapsLock пишем

"terminate:ctrl_alt_bksp,grp:caps_toggle,grp_led:scroll"

– **Установка консольного шрифта с кириллицей**

Редактируем файл /etc/rc.d/rc.font.

```
unicode_start Cyr_a8x16
for i in 1 2 3 4 5 6;do
echo -ne "\033%G" >/dev/tty$i
 done
```

**- lilo**

Редактируем файл /etc/lilo.conf ,
заменяем

```
append=" vt.default_utf8=0"
```

на:

```
append=" vt.default_utf8=1"
```

и выполняем команду:

```
/sbin/lilo -v
```

(не понадобилось, уже было vt.default_utf8=1)

**- Отображение русских имен файлов на разделах NTFS**

Редактируем файл /etc/fstab.
Пример

```
/dev/sda* /mnt/sdb5 ntfs-3g locale=ru_RU.utf8,umask=000 1 0
```

Русификация наверное единственный раздел в написанном (кроме разве что раздела с Xorg, если Патрик все же переедет на что то более новое), который может зависеть от версии слаки. В этом случае смотрим на форумах, например www.linux.org.ru/wiki/enРусификация_Slackware_13_c_utf8, или на linuxforum.ru, unixforum.org, на которых довольно быстро появляются советы по русификации очередной версии слаки.

# IV - Пакеты в Slackware - общие положения

Пакеты в Slackware используются как готовые, так и созданные собственноручно.

## 1 - Управление пакетами

Список всех установленных пакетов находится в /var/log/packages, оттуда же их можно удалить при помощи removepkg <пакет>. Готовый пакет устанавливаем через installpkg <пакет>, хотя корректнее делать upgradepkg --install-new <пакет>. Обновляем через upgradepkg <пакет>. Переустановка при каких то глюках upgradepkg --reinstall <пакет>.

Стандартный менеджер пакетов для операций с ними slackpkg. Все что можно о нем сказать уже написано в slackpkg —help и man slackpkg. Там все просто и понятно, и очень немного.

Посмотреть какому установленному пакету принадлежит нужный файл можно простым поиском по содержимому файлов в этой директории. Также просто определяется время установки всякого хлама, который из любознательности был поставлен - простой сортировкой по времени в файл-менеджере.

Хлам и не только, который был удален, ищем в /var/log/removed-packages, так что если слака отказывается делать "ку" смотреть надо там, чего такого "ненужного" поудаляли :-).

Ориентироваться в пакетах установочного DVD помогают файлики MANIFEST - все файлы во всех пакетах, FILE_LIST - все пакеты по категориям, PACKAGES.TXT - описания пакетов, которые есть и на самом DVD и в инете, например здесь — www.slackware.org.uk/slackware/slackware-13.37/slackware.

Разруливания зависимостей в слаке нет как класса, что с одной стороны очень хорошо, так как позволяет получить именно ту систему, которую хочется, с другой плохо для любителей щелкнув мышкой получить все сразу, и то что надо и еще массу приятных вещей, которые совершенно не нужны. К примеру PClinuxOS при установке IceWM подтянул не только qt4 (что еще можно объяснить,так как одна из графических конфигурялок IceWM написана с использованием qt, зачем непонятно, но так написана), но и googlearth. Если по каким либо причинам надо все же узнать зависимости пакетов то их можно установить при помощи утилиты slackdeptrack - darkstar.ist.utl.pt/slackware/addon/slacky/slackware-13.0/utilities/slackdeptrack/0.1.3/

Если при операциях с пакетами, компилировании и конфигурации что то происходит не так, то помним что все они применяются от рута.

## 2 - Использование готовых пакетов

В оригинальной слаке которую выпускает Патрик пакетов не так уж много, тем не менее пакеты с установочного DVD вполне способны удовлетворить базовые запросы как домашнего, так и среднего корпоративного пользователя. Но поскольку базовыми запросами дело правило не ограничивается, то на помощь в этом случае приходят собранные другими пакеты. Наиболее известные места где можно найти готовые пакеты

slacky.eu
slackfind.net
rlworkman.net/pkgs
connie.slackware.com/~alien/slackbuilds
slackers.it
www.teoxonline.com/utils/sse
slak.homelinux.org
www.z01.eu/slak
www.teoxonline.com/utils/sse
slakfinder.frattocchie.it/slak

Существует также множество основанных на Slackware дистрибутивов, из которых можно брать готовые пакеты, но не всегда (хотя и в большинстве случаев) такие пакеты корректно работают. Лучше всего в этом плане работают пакеты взятые из дистрибутива zenwalk (packages.zenwalk.org/?v=current), хуже всего (практически никогда не работают) из дистрибутивов salix и absolute.

Пакеты с этих сайтов надо скачивать и устанавливать вручную. Можно автоматизировать этот процесс и даже использовать разрешение зависимостей при помощи утилиты slapt-get и ее фронтенда gslapt, но в этом случае надо внести все скомпилированные самим пакеты в список исключений, иначе при обновлении они затрутся стандартными версиями. Вообще говоря это не "путь Slackware" поскольку накладывает помимо сказанного еще и дополнительные ограничения и тем самым значительно сужается феноменальная гибкость слаки в управлении пакетами. Кроме того в них немеряно "особенностей", в частности удалять пакеты с их помощью надо крайне осторожно, запросто может снести полсистемы. Для slapt-get и gslapt есть сервис оповещения об обновлениях - software.jaos.org/#slapt-update-service

Возможно преобразование пакетов rpm в пакеты слаки при помощи утилиты rpm2tgz. Найти пакеты rpm можно на pkgs.org, там же есть пакеты и других форматов. Также можно преобразовать пакеты deb из debian в пакеты слаки (за исключением всего связанного с python, поскольку у debain специфическое расположение его файлов). Удобно пользоваться PackageConverter - code.google.com/p/foxoman/wiki/PackageConverter, GUI к alien, утилите преобразования пакетов разных форматов. Необходимы пакеты - сам alien и fakeroot со slackfind.net и dpkg со slackbuilds.org.

Если какая то программа после установки не запускается, смотрим вывод ее запуска в терминале, и ищем нужный пакет. Найти нужный пакет слаки по входящему в нему файлу можно на slak.homelinux.org или на www.z01.eu/slak, но он не всегда корректно работает, поэтому в этих случаях можно использовать поиск файлов по пакетам дебиана, www.debian.org/distrib/packages#search_packages, или по пакетам ubuntu, packages.ubuntu.com, для нахождения нужного пакета по аналогии. Поскольку в слаке, в отличие от того же дебиана и пр, зависимости минимальные и пакет объединяет в себе большинство необходимого для его запуска, то доставлять приходится как правило довольно мало.

При желании всегда можно перекомпилировать нужный пакет со своими опциями конфигурирования и флагами компиляции, как правило где то рядом с готовым пакетом в инете лежит его слакбилд, обычно в директории src или source.

## 3 - Создание пользовательских пакетов

Одна из изюминок слаки, за которую ее многие любят, это возможность довольно просто и быстро создавать свои пакеты, заточенные именно под свой комп и с теми возможностями, которые необходимы именно себе. Свои пакеты создаются из сорцов при помощи slackbuilds или традиционным методом компиляции из сорцов через конфигурацию - компиляцию и установку (make &&make install) - создание пакета, в ходе которого создается пакет.

При создании пакетов внимательно читаем файлы README, INSTALL и им подобные в папке с сорцами, зачастую в них кроме тривиальщины содержится крайне полезная информация, без которой программа просто не собирается.

Далее все команды выполняем или в папке с распакованными сорцами или в папке со слакбилдом.

Очень удобно при создании пользовательских пакетов пользоваться связкой worker-roxterm-juffed-набор скриптов, созданных для конфигурирования, компиляции и создания пакета. Такая связка приведена во вложении worker.tbz.

### - Что такое slackbuild и как их использовать

Slackbuild это скрипт с набором правил компиляции программы из сорцов, который создает пакет для установки. Этот пакет после выполнения скрипта прога.SlackBuild создается в /tmp (это в общем и наиболее распространенном случае, место зависит от переменной PKG= в slackbuild).

Наиболее известное место нахождения slackbuild - slackbuilds.org. Слакбилды для некоторых программ, которые есть в этом описании, но для которых нет слакбилдов на slackbuilds.or, можно найти здесь - www.wuala.com/SergMarkov19/Slackbuilds. К slackbuild нужны сами сорцы, линки на которых есть на slackbuilds.org. Сорцы кладем в директорию со slackbuild. Если на сайте программы есть более новая версия, чем указано в slackbuild, то со сайта скачиваем архив с сорцами новой версии и кладем его в папку со slackbuild, в котором правится VERSION=. Если хотите увековечить себя в имени пакета, правим в slackbuild переменную BUILD, вообще то делать так рекомендуется, чтобы затем отличить свой доморощенный самосбор от высочайше одобренного Патриком :-). Для настройки программы под свои вкусы в slackbuild правим опции configure (опции смотрим через ./configure --help в в папке с сорцами). Если на slackbuilds.org нет проги для нужной версии слаки, еще не собрали или вообще не собрали, то вполне можно взять слакбилд от предыдущей версии (или вообще в какой эта прога есть), скачать новые сорцы и изменить VERSION в слакбилде (и еще кое что подправить если потребуется), как правило такой метод работает.

Если же слакбилда нужной программы вообще нет и не хочется идти традиционным путем компиляции из сорцов через make &&make install — создание пакета, то писать свой slackbuild проще легкого, читаем FAQ - slackbuilds.org/faq и пишем свой слакбилд. Еще проще создать свой слакбилд со стандартными опциями и параметрами можно с помощью генератора слакбилдов на alien.slackbook.org/AST.

По аналогии со средствами автоматизации для бинарных пакетов есть подобные утилиты и для сборки пакетов из слакбилдов - slapt-src, sourcery, sbopkg. Но их использование не позволяет без самопальных костылей задать ни свои опции конфигурации программы ни свои оптимизационные флаги компиляции, и по сути собранные ими пакеты ничем не отличаются от уже собранных

бинарников.

Естественно можно пересобрать уже имеющиеся пакеты использую слакбилды для них. Для официальных пакетов слакбилды находятся в папке source/прога, для остальных где то примерно там же, в папках src, source и им подобных.

**- Где искать проги и брать их сорцы для создания пакетов**

На сайте конкретной программы, если знаете что именно надо. Не знаете какую программу хочется прямой путь на freshmeat.net, sourceforge.net. Если нашли какую то старую программу, которая понравилась, но сайт который уже давно в глубоком дауне и сорцов не сыскать, то в этом случае как правило помогает поиск таких сорцов на www.debian.org/distrib/packages или packages.ubuntu.com/ru, где есть и сорцы и патчи к ним, приспосабливающие старые сорцы под новые компилятор и либы.

Наиболее свежие версии программ распространяются через различные системы управления версиями, среди который наиболее распространены svn, git и mercurial. Такие свежие версии наиболее полезны для программ, которые привязаны к каким либо внешним интерфейсам, типа youtube и других подобных сайтов, поскольку изменения интерфейса к ним зачастую отражены только в самой новой версии программы из систем управления версиями. Естественно они могут быть полезны и в других случаях, например очень понравилось какая то версия программы, изменения в которой далее не были включены в официальные релизы, такое тоже бывает :-)

Для выкачивания последней версии сорцов из svn набираем в терминале

```
svn co (или checkout) <URL svn>
```

для выкачивания какой то определенной версии

```
svn co -r <номер ревизии> <URL svn>
```

Для выкачивания последней версии сорцов из git набираем в терминале

```
git clone <URL git>
```

Для выкачивания последней версии сорцов из mercurial набираем в терминале

```
hg clone < URL mercurial>
```

**- Флаги компиляции**

Настройка компилятора при создании пакетов из сорцов естественно имеет значение, и нельзя сказать что настройки по умолчанию в gcc являются оптимальными. Настройка осуществляем путем задания флагов компиляции.

Есть отличие в флагах компиляции для системных программ и пользовательских программ. Если первые, в число которых входят утилиты linuxbase (tar, bz2, zlib и.т.п), тулкиты (GTK, QT и другие), лучше собирать с принятыми в слаке флагами по умолчанию ("-O2 -march=i486 -mtune=i686"), то вторые лучше собирать с оптимизационными флагами. Также с флагами по умолчанию лучше собирать программы на python и perl.

Приведенные далее оптимизационные флаги определены не путем умозрительных заключений и теоретических представлений (таким образом почти все флаги оптимизации будут только улучшать, на практике это далеко не так, вернее совсем не так :-), эти флаги были определены путем тестирования производительности собранных пакетов. Ниже приведены примеры таких флагов для x86 (Ваши флаги скорее всего будут отличаться только поддержкой SSE).

Место задания флагов определяется способом конфигурирования программы. При конфигурации autoconf (в сорцах есть файл configure) флаги компиляции определяются в ~/.bashrc или в опциях configure, причем флаги в configure перекрывают флаги в ~/.bashrc. При конфигурации cmake (в сорцах есть файл CmakeLists.txt) флаги определяются в /usr/share/cmake-2.8/Modules/Compiler/GNU.cmake. При конфигурации qmake флаги определяются в /usr/lib/qt/mkspecs/common/g++.conf. В слакбилдах задание флагов компиляции определяется также исходя из способа конфигурирования, либо в секции configure либо в соответствующих файлах cmake и qmake.

**- Задание флагов компиляции в ~/.bashrc**

Редактируем файл ~/.bashrc

```
# gcc
export CHOST="i686-pc-linux-gnu"
export CFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1"
export CXXFLAGS="${CFLAGS}"
export LDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

где:

- O3 - уровень оптимизации
- march=native и mtune=native - настройка компилятора под Ваш процессор (mtune можно и не указывать, он оставлен для полноты) Некоторые программы (как правило системные а не прикладные) показывают более высокую производительность с -march=i686 и mtune=i686, даже если Ваш процессор намного мощнее.
- mmmx (m3dnow, msse) - поддержка соответствующих инструкций процессора (они не включаются флагом -march). Узнать поддерживаемые Вашим процессором инструкции можно через cat /proc/cpuinfo. Как именно их включать под свой процессор смотрим здесь gcc.gnu.org/onlinedocs/gcc-4.4.4/gcc/i386-and-x86_002d64-Options.html#i386-and-x86_002d64-Options.
  Пример: -msse(1,2,3,4) включает поддержку sse, -mno-sse(1,2,3,4) выключает
- pipe - отключение создания временных файлов при компиляции и передача данных напрямую, убыстряет компиляцию не в ущерб качеству.
- fomit-frame-pointer - не сохранять указатель на кадр (frame pointer) в регистре для функций, которые не нуждаются в этом. Это позволяет избежать инструкций на сохранение, определение и восстановление указателя на кадр (frame pointer); в то же время освобождая регистры для других функций. Это делает невозможным отладку на большинстве машин.

- falign* - не выравнивать по границам блока (актуально для процессоров включая и выше PIII и K6-2).
- Wl,-O1 -Wl,--as-needed - уменьшает количество требуемых библиотек для линковки (лучше не применять при компиляции системных пакетов).

Немного о флаге -O3, этот флаг оптимизационный, на одних программах он может улучшить производительность, на других, напротив, ухудшить. Но последнее верно в части весьма узкого круга программ, по крайней мере далее рассмотренные программы лучше работают именно с -O3.

Спорный флаг -ffast-math что называется «на любителя», особого выигрыша в производительности нет, но есть проблемы при сборке всего использующего sql(lite) .

Всегда помним базовый принцип оптимизации - не бывает много хорошей оптимизации, бывает только оптимальная оптимизация :-) , иначе, повключав множество разных флагов, почти гарантированно получаем только ухудшение, а не улучшение.

Приведенные флаги оптимизационные и при сборке всех программ, про которые говорится далее, они работают, но может случится и так, что какая то программа с этими флагами может собираться некорректно. Сначала узнаем собирается ли программа вообще - убираем LDFLAGS и заменяем CFLAGS на стандартный "-O2 -march=i486 -mtune=i686" Если программа собирается то далее заменяем последовательно в оптимизационных настройках -O3 на -O2, далее убираем вообще LDFLAGS, если и это не помогает убираем опции -falign и -fomit-frame-pointer.

Оптимизационные флаги используем только при сборке пользовательских программ, при сборке системных пакетов типа qt, gtk, zlib и.т.п и программ на питоне используем стандартные флаги

```
SLKCFLAGS="-O2 -march=i486 -mtune=i686"
и без LDFLAGS
```

После изменения флагов в ~/.bashrc не забываем перезагрузить сам bash, чтобы изменения вступили в силу (проще всего перезагрузить иксы).

Многие программы при компиляции через configure-make-make install используют свои флаги компиляции из Makefile, создаваемом в ходе configure. Если Вы твердо уверены что эти флаги лучше изменить (в большинстве случаев, кроме например mplayer и системных прог, так и есть, добавляем в конец опций Вашего configure

```
CFLAGS="-O3 -march=native -mtune=native -pipe \
-fomit-frame-pointer   -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native -pipe \
-fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1" \
LDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

Настройки в ~/.bashrc действуют при компиляции с использованием make &&make install, в slackbuild своя настройка флагов компиляции, которая перекрывает настройки в /root/.bashrc, поэтому каждый slackbuild правим под свой комп.

– **Задание флагов компиляции в configure**

Ниже приведен шаблон конфигурации при помощи configure с заданием в нем флагов компиляции

```
#!/bin/bash
#
./configure \
--build=i686-pc-linux-gnu \
--host=i686-pc-linux-gnu \
--prefix=/usr \
--libdir=/usr/lib \
--localstatedir=/var \
--sysconfdir=/etc \
--mandir=/usr/man \
CFLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
 -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 \
 -falign-labels=1" \
LDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

**- Задание флагов компиляции при использовании cmake**

Заменяем в файле (здесь нет " \" для перевода строки и все что начинается с "set" это на самом деле одна команда в одной строке )

/usr/share/cmake-2.8/Modules/Compiler/GNU.cmake

```
# Initial configuration flags.

set(CMAKE_${lang}_FLAGS_INIT "")
set(CMAKE_${lang}_FLAGS_DEBUG_INIT "-g")
set(CMAKE_${lang}_FLAGS_MINSIZEREL_INIT "-Os -DNDEBUG")
set(CMAKE_${lang}_FLAGS_RELEASE_INIT "-O3 -DNDEBUG")
set(CMAKE_${lang}_FLAGS_RELWITHDEBINFO_INIT "-O2 -g")

set(CMAKE_${lang}_CREATE_PREPROCESSED_SOURCE "<CMAKE_$
{lang}_COMPILER> <DEFINES> <FLAGS> -E <SOURCE> > <PREPROCESSED_SOURCE>")

set(CMAKE_${lang}_CREATE_ASSEMBLY_SOURCE "<CMAKE_${lang}_COMPILER>
<DEFINES> <FLAGS> -S <SOURCE> -o <ASSEMBLY_SOURCE>")

if(NOT APPLE)
```

на

```
# Initial configuration flags.
```

```
        set(CMAKE_${lang}_FLAGS_INIT "-O3 -march=native -mtune=native -pipe  -fomit-
frame-pointer -mmmx -m3dnow -falign-jumps=1 -falign-labels=1")

        set(CMAKE_${lang}_FLAGS_DEBUG_INIT "-g")
        set(CMAKE_${lang}_FLAGS_MINSIZEREL_INIT "-Os -DNDEBUG")
        set(CMAKE_${lang}_FLAGS_RELEASE_INIT "-O3 -march=native -mtune=native -pipe
-fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 -falign-labels=1")

        set(CMAKE_${lang}_FLAGS_RELWITHDEBINFO_INIT "-O3 -march=native
-mtune=native -pipe  -fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 -falign-
labels=1")
        set(CMAKE_${lang}_CREATE_PREPROCESSED_SOURCE "<CMAKE_$
{lang}_COMPILER> <DEFINES> <FLAGS> -E <SOURCE> > <PREPROCESSED_SOURCE>")

        set(CMAKE_${lang}_CREATE_ASSEMBLY_SOURCE "<CMAKE_${lang}_COMPILER>
<DEFINES> <FLAGS> -S <SOURCE> -o <ASSEMBLY_SOURCE>")

        if(NOT APPLE)
```

**- Задание флагов компиляции при использовании qmake**

Заменяем в файле /usr/lib/qt/mkspecs/common/g++.conf, CFLAGS в одной строке

```
QMAKE_CFLAGS          += -pipe
```

и

```
QMAKE_CFLAGS_RELEASE     += -O2 -march=i486 -mtune=i686
```

на

```
QMAKE_CFLAGS+= -O3 -march=native -mtune=native -pipe
-fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1
-falign-labels=1
```

и

```
QMAKE_CFLAGS_RELEASE     += -O3 -march=native -mtune=native \
-pipe   -fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1
```

В qt версий больше 4.8 проводим аналогичные замены в

/usr/lib/qt/mkspecs/common/gcc-base.conf и /usr/lib/qt/mkspecs/common/g++-unix.conf

**- Задание флагов компиляции в slackbuild**

**- при использовании configure**

Во всех slackbuild (кроме системных пакетов) заменяем

```
if [ "$ARCH" = "i486" ]; then
SLKCFLAGS="-O2 -march=i486 -mtune=i686"
LIBDIRSUFFIX=""
elif [ "$ARCH" = "i686" ]; then
SLKCFLAGS="-O2 -march=i686 -mtune=i686"
LIBDIRSUFFIX=""
elif [ "$ARCH" = "x86_64" ]; then
SLKCFLAGS="-O2"
LIBDIRSUFFIX="64"
else
SLKCFLAGS="-O2"
LIBDIRSUFFIX=""
fi
```

на (для примера) -только для x86 (флаги естественно подставляем свои, по аналогии с ~/.bashrc)

```
if [ "$ARCH" = "i486" ]; then
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
-fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "i686" ]; then
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
-fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "x86_64" ]; then
  SLKCFLAGS="-O2"
  LIBDIRSUFFIX="64"
else
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
-fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
  LIBDIRSUFFIX=""
fi
SLKLDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

и заменяем также в зависимости от того, что в слакбиде написано
если написано

```
CFLAGS="$SLKCFLAGS" \
```

то заменяем на

```
CFLAGS="$SLKCFLAGS" \
```

```
LDFLAGS="$SLKLDFLAGS" \
```

если написано

```
export CFLAGS="$SLKCFLAGS"
```

то заменяем на

```
export CFLAGS="$SLKCFLAGS"
export LDFLAGS="$SLKLDFLAGS"
```

Простейший скриптик для первого случая, здесь нет " \" для перевода строки и все что начинается с "sed" это на самом деле одна команда в одной строке

```
#!/bin/bash
# Скрипт заменяющий стандартные опции компилятора в слакбилдах со slackbuikds.org
# на свои опции
cp $1 $1.orig

sed -i 's/ SLKCFLAGS="-O2 -march=i486 -mtune=i686"/  SLKCFLAGS="-O3 -march=native
-mtune=native -fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 -falign-
labels=1"/g' $1

sed -i 's/ SLKCFLAGS="-O2 -march=i686 -mtune=i686"/  SLKCFLAGS="-O3 -march=native
-mtune=native -fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 -falign-
labels=1"/g' $1

sed -i 's/ SLKCFLAGS="-O2"/  SLKCFLAGS="-O3 -march=native -mtune=native -fomit-frame-
pointer -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1"/g' $1
sed -i 's/TAG=${TAG:-_SBo}/TAG=${TAG:-_am}/g' $1

sed -i 's/^NUMJOBS=.*/NUMJOBS=${NUMJOBS:-" -j2 "}/g' $1

sed -i 's/CFLAGS="$SLKCFLAGS" \\/CFLAGS="$SLKCFLAGS" \\\nLDFLAGS="-Wl,-O1 -Wl,--as-
needed" \\/g' $1

sed -i 's/\/sbin\/makepkg -l y -c n $OUTPUT\/$PRGNAM-$VERSION-$ARCH-$BUILD$TAG.$
{PKGTYPE:-tgz}/\/sbin\/makepkg -l y -c n $CWD\/$PRGNAM-$VERSION-$ARCH-$BUILD$TAG.
${PKGTYPE\:-txz}/g' $1
echo 'Опции компилятора исправлены'
```

**- при использовании cmake**

Точно также как и для случая configure но вместо export ставим как часть cmake

```
cmake \
  -DCMAKE_C_FLAGS:STRING="$SLKCFLAGS" \
  -DCMAKE_CXX_FLAGS:STRING="$SLKCFLAGS" \
```

**- при использовании qmake**

Точно также как и для случая configure но вместо export ставим как часть qmake

```
qmake \
  QMAKE_CFLAGS="$SLKCFLAGS" \
  QMAKE_CXXFLAGS="$SLKCFLAGS"
```

**- Задание флагов компиляции при использовании prelink**

Немного о prelink, который осуществляет предварительную линковку разделяемых библиотек и тем самым увеличивает производительность и особенно время запуска программ. В слаке 13.37 он не работает, но вполне возможно что будет работать в будущих версиях, как ранее работал в 13.1. Чтобы prelink работал со своими скомпилированными пакетами надо в опции CFLAGS и SLKCFLAGS добавить "-fPIC" и/или добавить в опции его configure --with-pic.

**- Компилирование с обратной связью**

Если есть много времени, можно попробовать использовать такую возможность gcc, как компилирование с обратной связью, при котором сначала создается исполняемый файл с дополнительным кодом, который затем при исполнении анализирует выполнение программы и записывает результата анализа в отдельный файл, а потом при повторной компиляции этот файл используется компилятором с целью оптимизации вновь создаваемого исполняемого файла (профилировщик+оптимизатор). Затраты времени на создание пакета при этом возрастают минимум раза в три. Выигрыш в производительности, это как повезет:-)

Общая схема использования:

- сначала компилируем с флагами

```
-O3 -march=native -mtune=native -pipe -mmmx -m3dnow \
-falign-jumps=1 -falign-labels=1 \
-fprofile-dir=/tmp/prof/proga  -fprofile-generate \
-fprofile-arcs -fprofile-values
```

где директория /tmp/prof/proga должны быть доступна для записи

- запускаем несколько раз программу с выполнением типичных для нее операций

- затем компилируем во второй раз с флагами

```
-O3 -march=native -mtune=native -pipe -mmmx -m3dnow \
-falign-jumps=1 -falign-labels=1 \
-fprofile-dir=/tmp/prof/proga -fprofile-use \
-fbranch-probabilities -fvpt -funroll-loops -fpeel-loops -ftracer
```

тут есть флаги, которые вставляются fprofile-use автоматом и они включены только для гарантии.

**- Конфигурирование программ**

Конфигурирование программ позволяет как использовать необходимые возможности программы, не входящие в ее конфигурацию по умолчанию, так и избавиться от ненужных ее возможностей в конфигурации по умолчанию, тем самым облегчив и ускорив ее работу. Конфигурирование осуществляется заданием необходимых опций конфигурации. Методы задания опций зависят от применяемого метода конфигурирования.

Далее сказанное применимо как при конфигурировании программы при использование make - создание пакета, так и при использовании слакбилдов, в которых правится секция ./configure или другая секция конфигурации.

Если конфигурация прошла успешно, но при компиляции возникают какие то ошибки, смотрим весь вывод конфигурирования, чего не хватает и какие есть предупреждения.

**- Конфигурирование программ, использующих configure**

В папке с распакованными сорцами таких программ лежит файл configure. Выполняем в папке с сорцами

```
./configure --help
```

Далее смотрим опции и конфигурируем по желанию (и хотению :-)

```
./configure - опции
```

Если опций много и их неудобно размещать в одной строке, разбиваем их на несколько строк, добавляя в конец каждой строки кроме конечной
[пробел]\.

Не оставляем умолчальную опцию --prefix=/usr/local, ставим или в /usr или в /opt/пакет, оставьте /usr/local для всяких прог, собираемых для пробы и на скорую руку. Если в дальнейшем планируется использовать prelink, (который не работает в 13.37) то надо добавить опцию --with-pic, если она есть в выводе ./configure --help.

Помним базовый принцип конфигурации - не впихивать все подряд, и что нужно и то что сто лет не будет нужно. Если через сто лет приспичит какая то возможность ее через сто лет и надо сделать, прогу лишние возможности только утяжеляют со снижением производительности того, чем пользуемся каждый день, а не раз в сто лет.

В результате использования ./configure создается Makefile и если что то все таки не нравится, то тогда уже править его, ну или вообще ковырять сам configure.

Удобно использовать шаблон конфигурирования, куда добавляются нужные опции.

```
#!/bin/bash
#
```

```
./configure \
--build=i686-pc-linux-gnu \
--host=i686-pc-linux-gnu \
--prefix=/usr \
--libdir=/usr/lib \
--localstatedir=/var \
--sysconfdir=/etc \
--mandir=/usr/man \
CFLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
 -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 \
 -falign-labels=1" \
LDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

**- Конфигурирование программ, использующих cmake**

В папке с распакованными сорцами таких программ лежит файл CMakeLists.txt. Выполняем в папке с сорцами

```
cmake \
-DCMAKE_INSTALL_PREFIX=/usr \
-DLIB_INSTALL_DIR=/usr/lib \
-DMAN_INSTALL_DIR=/usr/man \
-DSYSCONF_INSTALL_DIR=/etc \
-DINCLUDE_INSTALL_DIR=/usr/include \
-DCMAKE_C_FLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
 -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1" \
-DCMAKE_CXX_FLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
 -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
```

Часто необходимо создать директорию build и уже из нее запускать cmake, об этом как правило написано в README или install, тогда

```
mkdir build
cd  build
cmake .. \
-DCMAKE_INSTALL_PREFIX=/usr \
-DLIB_INSTALL_DIR=/usr/lib \
-DMAN_INSTALL_DIR=/usr/man \
-DSYSCONF_INSTALL_DIR=/etc \
-DINCLUDE_INSTALL_DIR=/usr/include \
-DCMAKE_C_FLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
 -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1" \
-DCMAKE_CXX_FLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
 -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
```

У cmake есть интересная особенность, позволяющая исключать отдельные директории из папки проги при компиляции, для этого надо использовать опцию -DBUILD_foo=OFF, где foo соответствующая папка, которую вы хотите исключить при компиляции. При этом не надо опасаться что исключите что то нужное, если

такое произойдет то cmake проигнорирует эту опцию.

Для каких то объемных прог далее можно запустить ccmake .. и установить свои опции конфигурирования, но как правило это не требуются, да и зачастую авторы не предусматривают такую возможность.

**- Конфигурирование программ, использующих qmake**

В папке с распакованными сорцами таких программ лежит файл *.pro. Выполняем в папке с сорцами

```
qmake \
  QMAKE_CFLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
 -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1" \
  QMAKE_CXXFLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
 -pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
```

**- Конфигурирование программ, использующих waf**

Иногда, чаще всего для каких то гномовых прог, применяется создание пакета при помощи waf. В папке с распакованными сорцами таких программ лежит файл waf. Конфигурация:

```
./waf configure \
  --prefix=/usr \
  --mandir=/usr/man
```

**- Другие методы конфигурирования программ**

Иногда в папке с сорцами нет ни одного из вышеперечисленных файлов, но лежит файл autogen.sh. Запускаем этот файл и он создает файл configure. Далее как описано выше.

Для самых простых программ нет и этого, а есть только Makefile, тогда и кон фигурировать нечего, просто собираем пакет.

**- Компилирование и создание пакета**

После конфигурации программы остается ее скомпилировать и создать установочный пакет для слаки. Эти операции лучше рассматривать вместе.

Стандартный метод следующий (от рута)

```
make
 make install DESTDIR=/tmp/packages-temp
cd /tmp/packages-temp
makepkg /tmp/packages.tgz (или makepkg /tmp/packages.txz)
cd ..
rm -rf /tmp/packages-temp
```

packages.txz естественно заменяем на свое-имя-пакета.txz.

В случае использования qmake для конфигурации пакет создается следующим образом (от рута)

```
make
make install INSTALL_ROOT=/tmp/packages-temp
cd /tmp/packages-temp
makepkg /tmp/packages.tgz (или makepkg /tmp/packages.txz)
cd ..
rm -rf /tmp/packages-temp
```

packages.txz естественно заменяем на свое-имя-пакета.txz.

Такая схема сборки пакета работает, если в Makefile есть параметр DESTDIR, он как правило есть, но если в Makefile его нет, то и нет такой возможности и пакет собрать таким образом будет нельзя, поскольку пакет будут ставиться сразу в /usr а не в папку для создания пакета, также этот метод не работает если некорректно работает INSTALL_ROOT в пакете, сконфигурированным с использованием qmake. В этом случае остается только тихо (ну или громко :-) славя разработчиков такой прелести смотреть в секции Install в Makefile что и куда таким образом поставилось, удалять все это ручками, и применять один из способов ниже.
Первый способ - править Makefile.
В секции Install заменять что то типа

```
-install -d $(PREFIX)/bin/
```

на

```
-install -d $(DESTDIR)/$(PREFIX)/bin/
```

Второй способ - вместо makepkg использовать slacktrack, т.е. вместо

```
make install
cd /tmp/5
makepkg /tmp/пакет.tgz (или makepkg /tmp/пакет.txz)
```

использовать

```
slacktrack -cmrzSbY -p пакет.tgz make install
```

естественно нужно сначала установить пакет slacktrack с dvd.

Нежелательно сразу использовать slacktrack по простой причине, при работе он сканирует всю корневую файловую систему два раза, сначала как образец, второй для нахождения различий до и после установки, время это занимает довольно много. Если это не смущает лучше сразу использовать именно его. При его работе лучше не использовать проги обращающиеся к дискам. Или можно чуть модифицировать сам скрипт, заменив в /usr/bin/slacktrack

```
EXCLUDE_LIST="/dev/shm|/dev/shm/|/dev/shm$|/var/run/|/var/run$|/etc/dhcpc/|/etc/dhcpc
$|/var/cache/|/var/cache$|/media$|/media/|/srv$|/srv/|/selinux$|/selinux/|/var/lib/rpm|/var/li
```

```
b/rpm$|/var/yp$|/var/yp/|/sys$|/sys/|/initrd$|/initrd/|/dev/input$|/dev/input/|/dev/.udev/|/de
v/.udev$|/dev/vc$|/dev/vc/|/dev/console|/dev/pts$|/dev/pts/|/dev/ptmx|/dev/tty|/var/log|/et
c/mtab|/etc/resolv.conf|/etc/ld.so.cache|/tmp|/root|/proc|/var/tmp|/var/run/utmp"
```

на

```
EXCLUDE_LIST="/dev|/dev/|/dev$|/var/run/|/var/run$|/etc/dhcpc/|/etc/dhcpc$|/var/cache/|/
var/cache$|/media$|/media/|/srv$|/srv/|/selinux$|/selinux/|/var/lib/rpm|/var/lib/rpm$|/var/yp
$|/var/yp/|/sys$|/sys/|/initrd$|/initrd/|/dev/input$|/dev/input/|/dev/.udev/|/dev/.udev$|/dev/
vc$|/dev/vc/|/dev/console|/dev/pts$|/dev/pts/|/dev/ptmx|/dev/tty|/var/log|/etc/mtab|/etc/res
olv.conf|/etc/ld.so.cache|/tmp|/root|/proc|/var/tmp|/var/run/utmp"
```

при такой замене пакеты, что то создающие в /dev будут некорректно собираться, но как правило такие пакеты очень редки, поэтому можно сделать такую замену.

Также удобно использовать простейший скриптик для создания пакета, в котором пакет создается в текущей директории с именем пакета как именем директории

```
#!/bin/sh
#
ARCH=${ARCH:-i686}

# Директории, использущиеся при компиляции
CWD=$(pwd)
name=$(basename $(pwd))
TMP=${TMP:-/tmp/compile-am}
PKG=$TMP/package-$name
OUTPUT=${OUTPUT:-/tmp}

# Очистка предыдущей компиляции
rm -rf $PKG

# Флаги компиляции и сама компиляция
SLKCFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1"
CFLAGS="$SLKCFLAGS"
CXXFLAGS="$SLKCFLAGS"
LDFLAGS="-Wl,-O1 -Wl,--as-needed"

# nice -n 19 устанавливает самый низкий приоритет для make, если
# нужна более быстрая компиляция при которой make отбирает
# больше ресурсов и не дает нормально работать другим программ
# уберите nice -n 19
nice -n 19 make || exit 1
make install DESTDIR=$PKG || exit 1

# strip
find $PKG | xargs file | grep -e "executable" \
-e "shared object" | grep ELF \
| cut -f 1 -d : | xargs strip --strip-unneeded 2> /dev/null || true
# Сжать маны
```

```
if [ -d $PKG/usr/man ]; then
  find $PKG/usr/man -type f -name "*.?" -exec gzip -9f {} \;
  for i in $(find $PKG/usr/man -type l -name "*.?") ; \
do ln -s $( readlink $i ).gz $i.gz ; rm $i ; done
fi

# Копирование doinst.sh
mkdir -p $PKG/install
cat /opt/scripts/doinst.sh > $PKG/install/doinst.sh

# Создание готового пакета с makepkg
cd $PKG
/sbin/makepkg -l y -c n $CWD/$name.txz

# Создание при помощи slacktrack
# slacktrack -mzSp $CWD/$PRGNAM-$VERSION-$ARCH-$BUILD-$TAG.tgz \
# make install

#cp /tmp/$PRGNAM-$VERSION-$ARCH-$BUILD-$TAG.tgz \
#$CWD/$PRGNAM-#$VERSION-$ARCH-$BUILD-$TAG.tgz
#rm -rf /tmp/$PRGNAM-$VERSION-$ARCH-$BUILD-$TAG.tgz
echo ""
echo "Пакет" "$name.txz" "создан в рабочей директории"
```

Скрипт doinst.sh определяет послеустановочные процедуры и в самом общем случае выглядит так

```
config() {
  NEW="$1"
  OLD="$(dirname $NEW)/$(basename $NEW .new)"
  # If there's no config file by that name, mv it over:
  if [ ! -r $OLD ]; then
    mv $NEW $OLD
  elif [ "$(cat $OLD | md5sum)" = "$(cat $NEW | md5sum)" ]; then
    # toss the redundant copy
    rm $NEW
  fi
  # Otherwise, we leave the .new copy for the admin to consider...
}

preserve_perms() {
  NEW="$1"
  OLD="$(dirname $NEW)/$(basename $NEW .new)"
  if [ -e $OLD ]; then
    cp -a $OLD ${NEW}.incoming
    cat $NEW > ${NEW}.incoming
    mv ${NEW}.incoming $NEW
  fi
  config $NEW
}

schema_install() {
```

```
  SCHEMA="$1"
  GCONF_CONFIG_SOURCE="xml::etc/gconf/gconf.xml.defaults" \
  chroot . gconftool-2 --makefile-install-rule \
    /etc/gconf/schemas/$SCHEMA \
    1>/dev/null
}

schema_install blah.schemas
preserve_perms etc/rc.d/rc.INIT.new
config etc/configfile.new

if [ -x /usr/bin/update-desktop-database ]; then
  /usr/bin/update-desktop-database -q usr/share/applications >/dev/null 2>&1
fi

if [ -x /usr/bin/update-mime-database ]; then
  /usr/bin/update-mime-database usr/share/mime >/dev/null 2>&1
fi

if [ -e usr/share/icons/hicolor/icon-theme.cache ]; then
  if [ -x /usr/bin/gtk-update-icon-cache ]; then
    /usr/bin/gtk-update-icon-cache usr/share/icons/hicolor >/dev/null 2>&1
  fi
fi

if [ -x /usr/bin/glib-compile-schemas ]; then
  /usr/bin/glib-compile-schemas /usr/share/glib-2.0/schemas/
fi
```

Но в большинстве случаев, когда нет установки файлов конфигурации в /etc и и необходимости компиляции schemas для гномовых прог, вполне подходит его сокращенный тип

```
if [ -x /usr/bin/update-desktop-database ]; then
  /usr/bin/update-desktop-database -q usr/share/applications >/dev/null 2>&1
fi

if [ -e usr/share/icons/hicolor/icon-theme.cache ]; then
  if [ -x /usr/bin/gtk-update-icon-cache ]; then
    /usr/bin/gtk-update-icon-cache usr/share/icons/hicolor >/dev/null 2>&1
  fi
fi
```

В оригинальном виде скрипт создает пакет при помощи makepkg, если нужно создать пакет при помощи slacktrack читаем комментарии и закомментируем и раскомментируем соответствующие строки.

**- Компилирование и создание пакета для программ на python**

Довольно много прог на python, у которых своя процедура установки. В общем случае как всегда надо смотреть в INSTALL и/или README файлы в папке с сорцами, но в большинстве случаев установка из папки с сорцами производится

следующим образом, package.txz естественно заменяем на свое-имя-пакета.txz.

```
python ./setup.py build
python setup.py install --root=/tmp/python-package
cd /tmp/python-package
/sbin/makepkg  -l y -c n /tmp/package.txz
rm -rf /tmp/python-package
```

В общем случае запускаем python setup(install).py --help и смотрим что надо делать. В некоторых случаях можно напрямую запускать соответствующий файл из распакованного архива с сорцами.

**- Компилирование и создание пакета для программ на perl**

Довольно мало, но еще есть проги на perl, которые как правило требуют множество перловых модулей со CPAN. Ставить их вручную удовольствие ниже среднего, поэтому проще воспользоваться скриптом cpan2tgz, который можно найти на slackbuilds.org, пользоваться им просто

```
cpan2tgz --no-install --pkgdir=/tmp --build-tag=-am имя-модуля
```

Для работы cpan2tgz нужны модули
CPAN - www.cpan.org/authors/id/A/AN/ANDK и
Getopt-Long — www.cpan.org/authors/id/J/JV/JV.

Нужный пакет создается в /tmp, причем создаются и все требуемые для него пакеты, а не только он сам. Название нужного недостающего модуля смотрим в выводе терминала при запуске в нем программы на perl, если вывод «no blabla/ablabl» то имя-модуля скорее всего blabla::ablabl, в более общем случае смотрим на www.cpan.org.

Если собранный пакет чем то не устраивает, то делаем в папке с сорцами make clean для очистки результатов предыдущей компиляции, заново конфигурируем, компилируем и создаем новый пакет (хотя проще удалить папку с сорцами и по новой распаковать их из архива с сорцами :-)

**- Компилирование и создание пакета при помощи waf**

Общая схема создания пакета в данном случае:

```
./waf  build
./waf install —destdir=/tmp/packages-temp
cd /tmp/packages-temp
/sbin/makepkg -l y -c n packages.txz
```

где packages — имя пакета.

## 4 - Установка другой версии системного пакета

Допустим вы нашли какую то очень понравившуюся программу, но она работает только с более новой ( или более старой) версией некоторых пакетов, чем установлены в системе. Выход довольно простой, устанавливаем эти новые (старые) версии в место, отличное от уже имеющихся в системе пакетов, и указываем ей как надо использовать эти новые пакеты.

Далее на конкретном примере, ну очень понравился например evince третьей версии но он требует GTK3, а в системе установлен GTK2. Сначала создаем пакет GTK3 с установкой куда нибудь в /opt/system/gtk-3.1.2, конфигурируем его по образу и подобию слакбилда GTK2, начнет при конфигурации ругаться на неправильные опции, смотрим в ./configure --help что изменилось в сравнении с GTK2. Конфигурация что то типа такого, обязательно указываем в --prefix место отличное от /usr, лучше куда нибудь в /opt

```
./configure \
--build=i686-pc-linux-gnu \
--host=i686-pc-linux-gnu \
--prefix=/opt/system/gtk-3.1.2 \
--enable-xinput \
--enable-xkb \
--disable-introspection \
CFLAGS="-O2 -march=i486 -mtune=i686" \
CXXFLAGS="-O2 -march=i486 -mtune=i686"
```

Поскольку системный пакет, то применяем стандартные опции компиляции, а не оптимизированные. Создаем пакет GTK3, устанавливаем, для порядка сделаем от рута updatedb ; locate *.pc | grep gtk.

Создаем привязки к либам GTK3 и вносим их в кэш либов, дополняем /etc/ld.so.conf, добавляя в него путь к либам GTK3, должно получится что то типа такого

```
/usr/local/lib
/usr/i486-slackware-linux/lib
/usr/lib/seamonkey
/usr/lib/qt/lib
/opt/e17/lib
/usr/lib/xulrunner
/opt/system/gtk-3.1.2/lib
```

Выполняем от рута ldconfig, чтобы изменения вступили в силу.

Теперь указываем системе где искать GTK3 при компиляции программ, добавляем в configure соответствующей программы, в данном случае evince

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH: \
/opt/system/gtk-3.1.2/lib/pkgconfig/
```

Для evince должно получиться что то типа такого, ставить новый пакет будем не в /usr, где уже стоит старый пакет, а куда нибудь в /usr/local или в /opt, чтобы не мешал уже имеющемуся пакету.

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH: \
/opt/system/gtk-3.1.2/lib/pkgconfig/ \
./configure \
--build=i686-pc-linux-gnu \
--host=i686-pc-linux-gnu \
--prefix=/usr/local \
--disable-scrollkeeper \
--disable-nautilus \
CFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow \
-falign-jumps=1  -falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow \
-falign-jumps=1 -falign-labels=1"
```

для остальных программ по аналогии.

Создаем пакет evince3 и устанавливаем его. Теперь указываем системе где искать GTK3 при выполнении программ на нем. Пишем в /root/.bashrc и в /home/user/.bashrc

```
export XDG_DATA_DIRS=/usr/share:/usr/local/share:\
/opt/system/gtk-3.1.2/share
export XDG_SHARED_DIR=/usr/share:/usr/local/share:\
/opt/system/gtk-3.1.2/share
```

Перезагружаем иксы чтобы применить изменения в ~/.bashrc Далее фишка специфичная для GTK3, выполняем для компиляции схемы org.gtk.Settings.FileChooser.gschema.xml

```
glib-compile-schemas /opt/system/gtk-3.1.2/share/glib-2.0/schemas/*
```

Теперь можно запускать evince3 оттуда, куда вы его установили, и наслаждаться :-) Естественно это общая схема установки таких пакетов, работает она в большинстве случае, если есть какие то специфические особенности, ищем в гугле вылезающую в терминале ошибку, находится как правило за пять минут.

Очень удобно создание и управление пакетами реализовать через файл-менеджер worker, написав простейшие скрипты автоматизации создания пакетов и назначив их и команды управления пакетами на кнопки worker. Пример таких скриптов и конфигурации worker в связке с терминалом roxterm и редактором juffed во вложении worker.tbz. Естественно что скрипты можно использовать и сами по себе, но в связке worker — roxterm - juffed ими наиболее удобно пользоваться.

## V - Пересборка некоторых базовых системных пакетов

Осуществляется с целью повышения быстродействия и производительности. Такая переборка необязательна, но желательна, т.к. довольно сильно повышает производительность. Не хотите возиться с их перекомпиляцией, можно этого и не делать, но и «мгновенной» системы тогда не получите :-).

### 1 - Перекомпиляция ядра

Перекомпилируем под свой комп и применяем патчи к ядру, которые на десктопе значительно повышают производительность и выполнение многозадачности, поскольку оригинальные вещи в ядре заточены под серверные, а не под десктопные задачи. В результате создаем маленькое монолитное быстродействующее ядро для десктопа.

Все сказанное в части выбора и конфигурирования ядра, патчей к нему относится к любым версиям слаки, в том числе и к будущим, только внимательно следим чтобы патчи подходили под версию ядра, и если планируем ставить фирменные драйвера видеокарты, также чтобы выбранная версия ядра поддерживалась нужным драйвером видеокарты. Если драйвер nvidia не собирается с выбранным ядром, то ищем патчи для нужного драйвера на packages.ubuntu.com. Если таких патчей нет и нужен драйвер, то с этой версией ядра придется погодить, пока либо nvidia либо Марк не наваяют требуемое :-)

Патчи к ядру можно разделить на одиночные и комбинированные. К первым относятся патчи, изменяющие только одну функцию ядра. Примеры таких патчей — BFS (альтернативный планировщик задач), BFQ (альтернативный планировщик ввода-вывода), TuxOnIce (hibernate). Вторые включают в себя набор одиночных патчей. Примеры таких патчей — ck1 (включает BFS и ряд других патчей для десктопа), pf (включает BFS, BFQ, TuxOnIce, ряд других патчей и апдейты к ядру), zen (весьма широкий набор патчей). Эффективность применения патчей сильно зависит от конкретного компа, патчи BFS, BFQ, ck1 применимы ко всем компам, pf содержит специфические патчи для ноут- и нетбуков, zen патч универсальный, но именно в силу универсальности менее эффективный. В общем случае лучше всего подбирать одиночные патчи под свой комп исходя из требуемых задач и имеющегося железа, для настольного десктопа наиболее эффективно сочетание патчей ck1 и BFQ, для ноут- и нетбуков патч pf.

Дальнейшее применимо для ядра версии выше 2.6.37, причем как к версиям 2.** так и к версиям 3.** и скорее всего и к последующим.

Сорцы ядра - www.kernel.org/pub/linux/kernel. Берем базовую, а не минорную, версию ядра, то есть брать 2.6.37 а не 2.6.37.5, поскольку патчи делаются именно под базовую версию ядра. Немного о выборе версии ядра. Как правило для десктопа на том же железе более новое ядро тормознее чем старое, исключений из этого правила практически нет, поэтому если ваше железо поддерживается более старой версией ядра, не имеет смысла увлекаться более новыми версиями. В новье впиханы какие то новые серверные технологии, которыми вы на десктопе скорее всего никогда не будете пользоваться, но которые тормозят ядро именно для десктопа.

Распаковываем ядро в /usr/src и делаем симлинк от директории /usr/src/linux-*.**.** на /usr/src/linux, дальнейшие операции производим в директории /usr/src/linux.

Если ядро скачиваем с kernel.org, а не устанавливаем из пакета с DVD, то смотрим в пакете kernel-source-* с DVD файл ../install/doinst.sh и делаем то, что в нем сказано. В данном примере используются патч ck1 — ck.kolivas.org/patches и патч BFQ — algo.ing.unimo.it/people/paolo/disk_sched/patches. Естественно можно применить и другие патчи, исходя из своего железа и задач.

Перекомпиляция ядра вещь совершенно безопасная при выполнении нескольких простых шагов:

- Копируем куда-нибудь /boot, /lib/modules и /etc/lilo.conf. Если эксперименты не удовлетворяют, восстанавливаем /boot, /lib/modules и /etc/lilo.conf, выполняем lilo -v и получаем старую конфигурацию. Если напортачили так, что с винта вообще ничего не загружается, загружаемся с какого нибудь liveCD, восстанавливаем /boot, /lib/modules и /etc/lilo.conf, затем загружаемся с установочного DVD Slackware, внимательно читаем самый первый экран, пишем в строку приглашения
  hugesmp.s root=/dev/sda1 rdinit= ro
  где sda1 ваш раздел со слакой, пробел после rdinit= обязателен.
  Выполняем lilo -v. Можно сделать chroot с liveCD в раздел со слакой, но восстановление с установочного DVD гарантированно работает.

- Патчим ядро, patch -p1 -i патч, сначала патч ck1, затем все подряд по порядку нумерации bfq в папке с сорцами ядра /usr/src/linux.

- Заменяем /usr/src/linux/.config на распакованный из /proc/config.gz (не забываем переименовать распакованный config в .config :-)

- При необходимости подстраиваем конфиг от старой версии ядра к новой - make oldconfig.
  Здесь лучше жать Enter на умолчания.

- Убираем лишние модули - make localmodconfig

- Включаем необходимые модули в ядро и выключаем initrd - make localyesconfig и создаем таким образом маленькое монолитное быстродействующее ядро

-

- Конфигурируем ядро - make menuconfig -нужные опции ищутся через / Читаем вверху клавиши управления.

- Включаем BFQ как планировщик по умолчанию (BFS сам включается) Enable the block layer-IO shedulers-Default IO Sheduler- BFQ, убираем поддержку остальных планировщиков

- Задаем компрессию ядра - General setup-Kernel compression mode bzip2.

- Выбираем более лучшую чем стандарная систему управления памятью Choose SLAB allocator (SLUB (Unqueued Allocator)) — SLUB

- Отключаем возможности для нестандартных (малых) систем Configure standard kernel features (for small systems)

- Processor type and features
  - Tickless System (Dynamic Ticks) = off
  - Preemption Model Preemptible kernel (Low latency desktop)
  – Timer frequency 1000Hz (Можно поставить значение выше, но это не всегда оправдано, для большинства задач хватает и 1000, увеличение этого значения может потребоваться для специфических задач, пр уменьшении эффективности большинства других задач)

  - Ставим свой процессор
    Processor type and features - Processor family
  - Отключаем создание универсального ядра, нужное только для дистрибутивов (GenericX86 support) и Math emulation, нужное только для супердревних процессоров.
  – Ставим количество своих процессоров - Maximum number of CPU. Не ставьте 1 и не отключайте поддержку SMP даже если у вас один процессор, не будут собираться драйвера NVIDIA.

- Power management and ACPI options - CPU Frequency scaling — No. Только для настольного десктопа, для ноут- и нетбуков необходимо включить и выбрать тип userspace.

- Включаем поддержку групп (General Setup-Control group support) и все что к ней относится во включаемом.

- Включаем в ядро (не как модуль) нужные файловые системы в File System.

- Включаем в ядро (не как модуль) нужные кодировки (CP437, 850, 855, 1250,ASCII, 866, CP1251, KOI-8R, UTF-8, ISO-8859-1, ISO859-5), File system - Native Language support. Задаем умолчальную (Default) кодировку — UTF-8.

- Если планируется использовать фирменный драйвера Nvidia то удаляем драйвера framebuffer для nvidia и rivaTNT, иначе фирменный драйвер не будет собираться.

- Убираем initrd (не нужен так как модуль файловых систем уже в ядре)
  - General Setup- Initial RAM filesystem and RAM `disk`

- Проверяем включены ли драйвера нужных контроллеров жестких дисков в ядро - Device Drivers - Serial ATA and Parallel ATA drivers, также проверяем включение нужных файловых систем в ядро - File systems. На всякий случай, `make localmodconfig` и `make localyesconfig` должны это сделать автоматом, но проверить не мешает, иначе система просто не загрузится.

- Убираем виртуализации - Virtualization

- Сохраняем конфигурацию - Save Alternative Configuration File (как .config естественно).

- Правим EXTRAVERSION в /usr/src/linux/Makefile, ставим что то свое, хоть .19:-)

- Временно перемещаем куда-нибудь ~/.bashrc, хоть в /tmp, его настройки при компиляции ядра только вредят.
- Собираем ядро - make bzImage

- Собираем модули - make modules

- Добавляем в lilo.conf (естественно заменяем раздел на свой)

```
# Linux bootable partition config begins
  image = /boot/vmlinuz.old
  root = /dev/sda*
  label = Slackold
# Partitions should be mounted read-only for checking
  read-only
```

  vmlinuz.old создается автоматом при установке нового ядра,lilo его прописывает при установке нового ядра, поэтому напортачив в конфигурации нового ядра всегда можно будет загрузить старое.

- Устанавливаем модули - make modules_install

- Устанавливаем новое ядро - make install

- Выполняем lilo -v

- Возвращаем на место /root/.bashrc и /home/user/.bashrc

- Перезагружаемся, выбираем новое ядро в lilo, вуаля :-)

Выше была описана простейшая конфигурация ядра, если собираем ядро в первый раз именно с нее рекомендуется начать, чтобы в ходе последующих экспериментов иметь возможность просто откатиться к уже имеющемуся оптимизированному ядру. В принципе для большинства случаев такой оптимизации вполне достаточно, но если хочется еще улучшить ядро, то сначала читаем публикацию "Ставим ядро 2.6, или Ядерная физика для домохозяйки. Версия 2.0" , которая с полпинка находится в инете (например здесь - www.ru.j-npcs.org/docs/add04/kernel-2.6-install-2.0.html). Она немного устарела, но основное в части описания параметров конфигурации ядра в ней есть, в том числе и для будущих версий ядра, третьего в том числе.

Дальнейшая оптимизация заключается в выкидывании ненужного из ядра. В принципе все основное что не нужно уже было выкинуто при помощи make localmodconfig, но при желании всегда можно найти чего еще выкинуть, главное в этом не перестараться :-). Естественно что критерии ненужности у каждого свои, здесь приведено только то, что можно безбоязненно удалить, если они вам не нужны. В дальнейшем подразумевается что уже умеем находить нужные параметры поиском через "/". Так как собирается монолитное ядро, то все необходимые составляющие включаются напрямую в ядро (*), а не как модуль (m).

Убираем всякие трассировки и отладки (trace, debug). Убираем поддержку лишних файловых систем и партиций. Далее можно аккуратно убрать лишнее из Net - поддержку IPv6, iptables (если нет параноидальности:-).

Убираем всякие хаки ядра (Kernel hacking), оставляем только Enable deprecated logic, Enable __must_check logic, Magic SysRq key, Strip assembler-generated symbols during link, Filter access to /dev/mem, Enable verbose x86 bootup info messages, Allow gcc to uninline functions marked 'inline'. Остальное это вообще то не хаки, а фичи для разработчиков, которые им нужны для отладки ядра, но которые только тормозят его работу пользователей.

Можно много чего убрать ненужного из ядра, но НЕ НАДО изменять значения параметров других настроек ядра, для этого надо иметь не начальные а глубокие знания по его устройству, если их нет (и не надо обольщаться что они есть :-), то не надо их трогать, лучше не станет, хуже станет запросто.

Для простоты переустановки ядра можно создать пакет с ним простейшим скриптиком

```
#!/bin/sh
mkdir -p /tmp/kernel-package/boot
cd /tmp/kernel-package
cp /boot/vmlinuz ./boot/
cp /boot/System.map ./boot/
# Здесь и далее **-** та часть, которая относится именно к этому ядру.
# Посмотреть какая именно проще по тому что прописали
# в EXTRAVERSION в /usr/src/linux/Makefile
cp /lib/modules/**-**/build/.config ./boot/config
mkdir -p /tmp/kernel-package/lib/modules
cp -a /lib/modules/**-** ./lib/modules/
makepkg -l y -c n /tmp/kernel-**-**-i686.txz
rm -rf /tmp/kernel-package
```

## 2 - Установка драйверов NVIDIA

- Ставим пакет dvd/extra/xf86-video-nouveau-blacklist/xf86-video-nouveau-blacklist-noarch-1.txz для блокировки драйвера nouveau или просто создаем файл /etc/modprobe.d/BLACKLIST-nouveau.conf и пишем в него

```
# Do not load the kernel nouveau dri module, since it
# interferes with both
# the nv and binary nvidia drivers.
blacklist nouveau
```

- Выходим из иксов — Ctrl-Alt-BackSpace.

- ./Nvidia-* в папке с драйвером. В конце создаем xorg.conf. Владельцы карт ATI здесь что то делают :-)
- В терминале nvidia-settings, ставим нужное разрешение (остальное потом), записываем в xorg.conf.

- В терминале nvidia-xconfig --composite --no-logo --render-accel

- В /etc/X11/xorg.conf должно получиться что то типа такого (для карт nvidia)

```
# nvidia-settings: X configuration file generated by nvidia-settings
# nvidia-settings:  version 1.0  (buildmeister@builder63)  Tue Jul 13 13:32:36 PDT 2010

Section "ServerLayout"
   Identifier     "Layout0"
   Screen      0  "Screen0" 0 0
   InputDevice    "Keyboard0" "CoreKeyboard"
   InputDevice    "Mouse0" "CorePointer"
EndSection

Section "Files"
   FontPath        "/usr/lib/X11/fonts/misc/:unscaled"
   FontPath        "/usr/lib/X11/fonts/100dpi/:unscaled"
   FontPath        "/usr/lib/X11/fonts/75dpi/:unscaled"
   FontPath        "/usr/lib/X11/fonts/misc/"
   FontPath        "/usr/lib/X11/fonts/Type1/"
   FontPath        "/usr/lib/X11/fonts/Speedo/"
   FontPath        "/usr/lib/X11/fonts/100dpi/"
   FontPath        "/usr/lib/X11/fonts/75dpi/"
   FontPath        "/usr/lib/X11/fonts/cyrillic/"
   FontPath        "/usr/lib/X11/fonts/TTF/"
EndSection

Section "Module"
        #Load  "GLcore"    #should be removed/commented out
            Load           "dbe"
          SubSection     "extmod"
             Option         "omit xfree86-dga" # don't initialise the DGA extension
          EndSubSection
          Load           "extmod"
          Load           "type1"
          Load           "freetype"
          Load           "glx"
        #Load "dri"        #should be removed/commented out
EndSection

Section "ServerFlags"
   Option         "Xinerama" "0"
EndSection

Section "InputDevice"
   # generated from default
   Identifier     "Mouse0"
   Driver         "mouse"
   Option         "Protocol" "auto"
   Option         "Device" "/dev/psaux"
   Option         "Emulate3Buttons" "no"
```

```
    Option         "ZAxisMapping" "4 5"
EndSection

Section "InputDevice"
    # generated from default
    Identifier     "Keyboard0"
    Driver         "kbd"
EndSection
Section "Monitor"
    # HorizSync source: edid, VertRefresh source: edid
    Identifier     "Monitor0"
    VendorName     "Unknown"
    ModelName      "Samsung SyncMaster"
    HorizSync       30.0 - 81.0
    VertRefresh     56.0 - 60.0
    Option         "DPMS"
EndSection

Section "Device"
    Identifier     "Videocard0"
    Driver         "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName      "GeForce4 MX 440"
    Option "AllowGLXWithComposite" "True"
    Option         "RenderAccel" "True"
EndSection

Section "Screen"
    Identifier     "Screen0"
    Device         "Videocard0"
    Monitor        "Monitor0"
    DefaultDepth    16
    Option         "metamodes" "1920x1080_60 +0+0; 1280x1024 +0+0; 1024x768 +0+0;
800x600 +0+0; 640x480 +0+0"
          Option         "NoLogo" "True"
            #
            # BackingStore - по возможности сохранять в буфере изображение области,
            #перекрытой окном, для
            # ускорения последующего восстановления данной области
            Option         "BackingStore" "True"

            # TripleBuffer - направляет вывод графики в дополнительный буфер перед
            # выводом на экран.
            # Улучшает плавность вывода графики на экран, но увеличивает время
            # реакции на пользовательские события
            # Option "TripleBuffer" "True"

            # DamageEvents - отправка сообщений о необходимости перерисовки
            # области (Отключается при MultiGPU)
            Option "DamageEvents" "True"

    SubSection     "Display"
```

```
        Depth       16
        Modes       "1600x1200" "1280x1024" "1024x768" "800x600" "640x480"
    EndSubSection
EndSection

Section "Extensions"
    Option         "Composite" "Enable"
    Option "RENDER"    "Enable"
EndSection
```

Есть стародавние советы включить AGP Fast Write и SB в драйвере Nvidia. Но никакой пользы от них нет, увеличение производительности на уровне десятых долей процента, если вообще она не падает, зато запросто можно пойметь немеряно всяких глюков.

Если захотите после установки драйверов nvidia установить свободные драйвера, надо кроме деинсталляции самого драйвера (смотрим в нем —help) и обратных процедур с nouveau, переустановить mesa.

Далее при замене флагов SLKCFLAGS и SLKLDFLAGS ставим свои, как описано выше в разделе "Подготовка компиляции" (отличаться будут скорее всего только поддержкой SSE).

## 3 — libjpeg-turbo

Версия libjpeg с использованием sse и mmx.
Слакбид и сорцы берем с slackbuilds.org. Лучшая версия libjpeg-turbo-1.1.0
Правим слакбилд, заменяем

```
if [ "$ARCH" = "i486" ]; then
  SLKCFLAGS="-O2 -march=i486 -mtune=i686"
  else
LIBDIRSUFFIX=""
elif [ "$ARCH" = "i686" ]; then
  SLKCFLAGS="-O2 -march=i686 -mtune=i686"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "x86_64" ]; then
  SLKCFLAGS="-O2 -fPIC"
  LIBDIRSUFFIX="64"
  SLKCFLAGS="-O2"
  LIBDIRSUFFIX=""
fi
```

на, например, (какие опции SLKCFLAGS и SLKLDFLAGS ставить смотрим в разделе "Подготовка в компиляции").

```
if [ "$ARCH" = "i486" ]; then
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
-mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "i686" ]; then
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
```

```
-mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "x86_64" ]; then
  SLKCFLAGS="-O2 -fPIC"
  LIBDIRSUFFIX="64"
else
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
-mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
  LIBDIRSUFFIX=""
fi
SLKLDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

также заменяем

```
  CFLAGS="$SLKCFLAGS" \
  CXXFLAGS="$SLKCFLAGS" \
```

на

```
  CFLAGS="$SLKCFLAGS" \
  CXXFLAGS="$SLKCFLAGS" \
  LDFLAGS="$SLKLDFLAGS" \
```

## 4 - cairo

Графическая библиотека.
Сорцы - dvd://source/l/cairo
Производим аналогичную libjpeg-turbo замену в слакбилде.
Правим слакбилд, заменяем

```
if [ "$ARCH" = "i486" ]; then
  SLKCFLAGS="-O2 -march=i486 -mtune=i686"
  else
LIBDIRSUFFIX=""
elif [ "$ARCH" = "i686" ]; then
  SLKCFLAGS="-O2 -march=i686 -mtune=i686"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "x86_64" ]; then
  SLKCFLAGS="-O2 -fPIC"
  LIBDIRSUFFIX="64"
  SLKCFLAGS="-O2"
  LIBDIRSUFFIX=""
fi
```

на, например, (какие опции SLKCFLAGS и SLKLDFLAGS ставить смотрим в разделе "Подготовка в компиляции").

```
if [ "$ARCH" = "i486" ]; then
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
-mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
  LIBDIRSUFFIX=""
```

```
elif [ "$ARCH" = "i686" ]; then
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
-mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
  LIBDIRSUFFIX=""
elif [ "$ARCH" = "x86_64" ]; then
  SLKCFLAGS="-O2 -fPIC"
  LIBDIRSUFFIX="64"
else
  SLKCFLAGS="-O3 -march=native -mtune=native -pipe \
-mmmx -m3dnow -falign-jumps=1 -falign-labels=1"
  LIBDIRSUFFIX=""
fi
SLKLDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

также заменяем

```
    CFLAGS="$SLKCFLAGS" \
```

на

```
    CFLAGS="$SLKCFLAGS" \
    LDFLAGS="$SLKLDFLAGS" \
```

Аналогичным образом можно ( и нужно) перекомпилировать с заменой флагов компиляции и линковки qt и gdk-pixbuf2.

## VI - Создание пакетов базового набора программ

Далее описываются опции конфигурации при создании пакетов базового набора программ. Пропускается задание флагов компиляции и сборка пакетов, которые описаны выше.

Программы по возможности ставим в /usr, оставим /usr/local для сборки чего то на быструю руку для проверки. Для создания пакетов лучше отвести место на другом linux разделе с тематической структурой (чтобы не путаться потом). Если не приведены опции ./configure, то конфигурация стандартная, описанная выше для каждого типа. Slackbuild берется со slackbuilds.org. Если slackbuild правится вдобавок к сказанному выше в части флагов компиляции, то об этом будет сказано. Сорцы берем с сайта программы, а не со slackbuilds.org (кроме особо оговоренных случаев), затем в слакбилде правим VERSION. Если последняя версия с сайта не компилируется (и такое бывает) то берем предпоследнюю и.т.д. Если нет ни опций ни слакбилда, то используется шаблон конфигурации. Необходимые пакеты и либы ставим до компиляции основной программы из готовых пакетов (или создаются свои пакеты). Если в результате компиляции образуется единственный файл, то он помещается в /usr/local/bin.

Немного о версиях прог, причем сказанное касается как выбора версии для первоначальной установки так и дальнейшей замены уже имеющейся версии на долее новую. Зачастую в новые версии включается совершенно ненужный именно вам функционал, который тем не менее значительно утяжеляет прогу, увеличивает время запуска и потребление ресурсов. В этом случае может быть оправданным установка более старой версии проги и отказ от замены версии на более новую. Естественно что это должно определяться для каждого конкретного случая исходя из личных предпочтений.

Сначала ставим со slackbuilds.org некоторые пакеты, которые затем будут использоваться для многих других пакетов - vala, libtasn, libgnome-keyring, gnome-keyring, icu4c, libgee, enca, ORBit2, GConf, lua.

### 1 - Системные программы

**IceWM** - оконный менеджер, со slackbuilds.org

Слакбид IceWM правится как сказано выше в части флагов компиляции и линковки, далее об этом не упоминается а подразумевается, кроме особо оговоренных случаев.

Нужна либа imlib2, сорцы и слакбилд берутся со slackbuilds.org. Далее в терминале xwmconfig, выбираем в меню IceWM, выходим из иксов (Ctrl-Alt-BackSpace), startx и загружается icewm.

**Openbox** - оконный менеджер, со slackbuilds.org

Для некоторых его утилит понадобится fuse-python — sourceforge.net/projects/fuse/files/fuse-python, сборка пакета стандарная для питона.

Ставим по порядку со slackbuilds.org — openbox (ту же версию сорцов что указана на slackbuilds.org), openbox-themes, obconf, obmenu. Далее ставим obtheme со xyne.archlinux.ca/projects/obtheme (просто распаковываем в /usr/local/bin), в нем заменяем

```
#!/usr/bin/env python2
```

на

```
#!/usr/bin/env python
```

Для автоматической генерации меню можно использовать menumaker со со slackbuilds.org. Задать правила для окон приложений можно при помощи OBApps со slackbuilds.org.

К openbox нужна панель, тут на выбор
**tint2, lxpanel, bmpanel2** - панели со slackbuilds.org, fbpanel - fbpanel.sourceforge.net
К lxpanel нужны lxmenu-data и menu-cache со slackbuilds.org.
Берем версии сорцов, указанные на slackbuilds.org.
Для установки картинок на рабочий стол openbox можно воспользоваться одной из следующих утилит
**feh** со slackbuilds.org. Нужна либа giblib со slackbuilds.org
или
**nitrogen** со slackbuilds.org. Нужна либы libsigc++, glibmm, cairomm, pangomm, mm-common, atkmm, gtkmm, libglademm, gconfmm. Берутся со slackbuilds.org. Эти либы потребуются для многих программ, так что если даже не ставить сам nitrogen то либы лучше поставить.

**Enlightenment E16** - оконный менеджер
Сорцы E16 и сопутствующие - sourceforge.net/projects/enlightenment/files
Сам E16

```
./configure \
--build=i686-pc-linux-gnu \
--host=i686-pc-linux-gnu \
--prefix=/usr \
--libdir=/usr/lib \
--localstatedir=/var \
--sysconfdir=/etc \
--mandir=/usr/man \
--enable-sound=no \
--enable-modules=yes \
CFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer  -pipe -mmmx -m3dnow \
-falign-jumps=1 -falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow \
-falign-jumps=1 -falign-labels=1" \
LDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

Делаем симлинк из ~/.e16 в ~/.enlightenment
Далее ставятся **e16menuedit -** редактор меню**, e16doc** - помощь по E16, **ew16-themes** - темы, **e16keyedit** - редактор шорткеев, **epplets** - апплеты. Сорцы берутся там же. Апплеты не появляются в меню, их надо добавлять в меню самому.

В /etc/X11/xinit создаем файл xinitrc.e16 следующего содержания и делаем исполняемым

```
#!/bin/sh
# $XConsortium: xinitrc.cpp,v 1.4 91/08/22 11:41:34 rws Exp $
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/etc/X11/xinit/.Xresources
sysmodmap=/etc/X11/xinit/.Xmodmap
# merge in defaults and keymaps
if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi
if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi
if [ -f $userresources ]; then
    xrdb -merge $userresources
fi
if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi
# Start the window manager:
exec /usr/bin/e16
```

Далее через xwmconfig выбираем нужный WM

**Enlightenment E17** - оконный менеджер

Ставятся по порядку со slackbuilds.org - embryo, eina, eet, evas, ecore, edje, e_dbus, efreet, eeze, enlightenment.

После первого запуска обязательно архивируем ~/.e , поскольку при настройках они, зачастую слетают и восстанавливать все же лучше не "с нуля" :-)

**xcompmgr —** композитный менеджер окон для X11. Ставим с dvd://x. Поскольку заменить один оконный менеджер другим просто нельзя, но хочется иметь эффекты типа теней под окнами и эффектов меню то самый простой путь это использовать xcompmgr с другими менеджерами окон, такими как IceWm и openbox. Он также позволяет переложить задачу отрисовки графики на видеокарту.

**worker** - файловый менеджер

Феноменально удобный и быстрый классический двухпанельник, табы, встроенная поддержка архивов, ftp и многое другое. Настраиваем по самое некуда, настроить можно практически все, и через гуй. Не смотрите скриншоты на сайте проекта:-) , они не обновлялись уже много лет, теперь есть и поддержка TTF шрифтов и настройка цветов и многое-многое другое. Настраивать его довольно муторно, настройка хотя и простая через GUI, но занимает много времени, т.к приходится вручную прописывать ассоциации, создавать свое контекстное меню, настраивать под себя кнопки, зато затем очень удобно и феноменально быстро. Особенно удобно его использовать в связке с терминалом roxterm и редактором juffed. Пример настройки такой связки смотрим во вложении worker.tbz.

Есть на slackbuilds.org.
У worker есть один недостаток, прыгающие каждый раз в разное место диалоговые окна. Проблема решается установкой

**devilspie -** менеджер окон
Он позволяет принудительно задать многие параметры окон, в т.ч. расположение на экране окон определенного класса.
Берем пакет rpm для предпоследней мандривы с pkgs.org и преобразуем в пакет слаки с помощью утилиты rpm2tgz.
Нужна либа libwnck со slackfind.net

Для devilspie есть frontend gdevilspie, сорцы (питон) - code.google.com/p/gdevilspie/downloads/list.
Можно обойтись встроенными средствами IceWM по позиционированию окон, но тогда придется запускать worker с английским интерфейсом. В этом случае пишем в ~/.icewm/winoptions для каждого диалогового окна worker что то типа Rename.Worker.geometry: +500+290
Название окна и его класс узнаем через xprop.

Для любителей explorer-like файловых менеджеров есть SpaceFM - sourceforge.net/projects/spacefm, форк PCManFM с расширенными возможностями. Хотя автор пишет о ранней стадии разработки spacefm вполне стабилен. Конфигурация стандартная, можно разве что добавить опцию --disable-superuser-checks

**roxterm** - терминал.
roxterm единственный из "легких" терминалов, который позволяет без "костылей" открывать команды в новом табе из других программ , что очень удобно при совместной работе с worker, да и с другими файл-менеджерами
Слакбилд и сорцы со slackbuilds.org.
Есть похожий по возможностям терминал lilyterm — lilyterm.luna.com.tw, но в нем нет поиска и он менее удобен чем roxterm

**juffed** - редактор
Прекрасный легкий редактор со множеством возможностей, в том числе такой как блоковое выделение вертикальных столбцов, прекрасным поиском с представлением результатов поиска в дополнительном табе внизу основного окна. Он есть на slackbuilds.org, но там он собран без плагинов.
Сорцы — sourceforge.net/projects/juffed/files. Лучшая версия — 0.8.1.
Конфигурация juffed

```
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/usr
```

Конфигурация плагинов стандартная для qmake.

**ne** — великолепный консольный редактор, для тех кому неудобен vim или emacs. По крайней мере у ne ноги растут из 1993 года еще из Amiga и сам автор так объяснил его портирование на линукс и дальнейшее развитие.. - "If you have the resources and the patience to use emacs or the right mental twist to use vi then

probably ne is not for you. However, if you need an editor that (очень много вкусностей)" то обратите внимание на ne.. :-) Привычные биндинги и вообще управление в целом, управление через хоткеи, меню и команды (как в виме), полностью настраиваемое меню и хоткеи, мультиредактирование, подсветка и еще очень много других вкусностей. Он хорош не только сам по себе но и как хорошая замена внутреннего просмотрщика worker. Берем с ne.dsi.unimi.it. Также есть слакбилд на slackbuilds.org. Полное описание на www.emerson.emory.edu/services/editors/ne/Top.html

Пакет создается простейшим скриптом

```
./version.pl
mkdir -p /tmp/ne-packages/usr
make PREFIX=/usr
make PREFIX=/tmp/ne-packages/usr install
cd /tmp/ne-packages
/sbin/makepkg -l y -c n -p /tmp/ne.tgz
```

Поскольку автор ne в новой, только что вышедшей версии, сотворил какую то лабуду с меню (часть пунктов не работает, части нужных команд в меню просто нет) вот здесь подправленное ~/.ne/.menus pastebin.com/GEpEUHCB, заодно и что примерно должно быть в ~/.ne/.default#ap — pastebin.com/SYBeMQR5, а здесь подсветка для слакбилдов — pastebin.com/tbBgdaLi.

Есть также хороший редактор tea (tea-editor.sourceforge.net), со своим оригинальным подходом к элементам интерфейса и множеством интересных и нужных возможностей. Но есть и два "но", первое это русопятство автора в терминах интерфейса в стиле "азм глаголю веди (зырить, ладить и.т.п) и второе это оригинальный подход, когда все, помощь, локали, подсветки, компилируется в бинарник. Над первым хорошо поржать первую минуту, но с работать с этим неудобно, хочется привычных названий, второе увеличивает размер бинарника и время запуска проги.

Решение как всегда тривиальное, для первого берем в лапки qt-linguist и правим русскую локаль, приводя все эти зырить-ладить в нормальные, привычные термины, второе решается удалением ненужного, как самих файлов (английская помощь, другие локали, ненужные подсветки и еще по вкусу) так и их перечисления в rlvn.qrc.

На выходе получается хорошая прога с привычными терминами, с меньшим по размеру и быстрее запускающимся бинарником.

Русская локаль в привычных терминах для tea-33.1.0, правленный rlvn.qrc и исправленная помощь — www.wuala.com/SergMarkov19/Guide-pdf/tea-33.1.0-classic-mod.tar.bz2/. В папке origin оригинальный tea, в папке mod поправленный.

Естественно есть еще и **geany** и масса других редакторов со slackbuilds.org, но для тех кто мало занимается или вообще не занимается программированием, их возможности все же избыточны.

**qxmledit -** редактор XML файлов, может понадобится для ручного редактирования XML конфигов - code.google.com/p/qxmledit. Конфигурация стандартная для qmake. Или взять со slackfind.net.

**xneur** - переключатель клавиатуры.
Сорцы - xneur.ru/downloads
Конфигурация xneur

```
./configure \
--build=i686-pc-linux-gnu \
--host=i686-pc-linux-gnu \
--prefix=/usr \
--libdir=/usr/lib \
--localstatedir=/var \
--sysconfdir=/etc \
--mandir=/usr/man \
--with-pic \
--with-sound=no \
--without-xosd \
--with-gtk \
--with-spell=aspell \
CFLAGS="-O3 -march=native -mtune=native -pipe -mmmx \
-m3dnow -falign-jumps=1 -falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native -pipe -mmmx \
-m3dnow -falign-jumps=1 -falign-labels=1"
```

gxneur, гуй к xneur- конфигурация по шаблону.

**regexxer** - поиск файлов и содержимого в текстовых файлах с возможностью редактирования содержимого найденных текстовых файлов прямо в утилите.

Нужны либы - libsigc++, glibmm, cairomm, pangomm, mm-common, atkmm, gtkmm, libglademm, gconfmm. Берутся со slackbuilds.org. Эти либы потребуются для многих программ, так что если даже не ставить сам regexxer, то либы лучше поставить.
Сорцы -regexxer.sourceforge.net. Конфигурация по шаблону. Есть слакбилд на slackbuilds.org.

**docfetcher** - утилита поиска в документах различных офисных форматов.
docfetcher.sourceforge.net/en/index.html
Поскольку java просто запустить скрипт.

**recoll** - утилита поиска по многим форматам с индексированием, простыми и сложными параметрами поиска, предпросмотром найденных файлов. Использует мощнейший backend Xapian. Одним из преимуществ recoll является то, что Xapian не запущен постоянно и не жрет ресурсы все время а работает по запросу, его можно как запустить вручную так и засунуть по расписанию в cron. Со slackbuilds.org (версия сорцов та же что указана на slackbuilds.org).

Во всех пакетах, нужных для recoll, и в нем самом, **не меняем** стандартные флаги компилятора на оптимизационные.

Нужны пакеты wv, wv2 со slackfind.net, antiword, catdoc, exiftool, mutagen, pstotext, pychm, python2-chardet со slackbuilds.org

Также нужны пакеты

- unrtf со slackbuilds.org,но вместо сорцов unrtf со slackbuilds.org берутся сорцы с поддержкой кириллицы отсюда www.lesbonscomptes.com/recoll/unrtf/unrtf-0.22.2beta.tar.gz, соответ-

ственно правится VERSION в слакбилде unrtf.
– xapian-core со slackbuilds.org. Если комп ну очень старый, без поддержки ssl, то правим слакбилд, приводя секцию configure к виду

```
./configure \
  --prefix=/usr \
  --libdir=/usr/lib${LIBDIRSUFFIX} \
  --sysconfdir=/etc \
  --localstatedir=/var \
  --mandir=/usr/man \
  --disable-sse \
  --docdir=/usr/doc/$PRGNAM-$VERSION \
  --disable-static \
  --build=$ARCH-slackware-linux
```

**meld** - сравнение файлов и папок со slackbuilds.org. Для компиляции необходим rarian со slackbuilds.org.

Также для сравнения файлов может быть полезен diffuse — diffuse.sourceforge.net. Создание пакета стандартное для питона, но вместо setup.py ставим install.py.
Есть слакбилд на slackbuilds.org

**psensor** - температурный монитор.
Сорцы - wpitchoune.net/psensor/files

**trashCaN** — корзина.
www.richardneill.org/source.php#cn или www.richardneill.org/src

Прописывается на кнопку в worker. Удобно в вызове скрипта назначить корзину не в стандарте freedesktop а в папку /.trash, т.е. назначить на кнопку worker - sudo cn -t /.trash -f {A} и прописать cn в /etc/sudoers.

Есть также корзина trash-cli со slackbuilds.org, более навороченная но и более медленная.

**clipit -** менеджер буфера обмена, форк parcellite, позволяющий, определить неудаляемые автоматически пункты. В нем есть и такая интересная особенность как прямая вставка в нужное окно, без "вставить", только выбором нужного пункта в меню clipit, но она некорректно работает со всеми офисами, clipit в этом случае впадает в ступор и помогает только его убиение через htop и перезапуск, так что лучше этой возможностью не пользоваться.
Нужна утилита xdotool со slackbuilds.org.
Сорцы — sourceforge.net/projects/gtkclipit/files. Есть слакбилд на slackbuilds.org

Есть также прекрасный менеджер буфера обмена CopyQ — github.com/hluk/CopyQ, с возможностью запоминать не только текст но и изображения, создавать группы, в том числе для постоянного запоминания, и многими другими вкусностями. Но у него есть одно «но», он не ловит буфер выделения в прогах на xlib, поэтому трудно его использовать с такими прогами как например worker.

Впрочем при использовании cairo-dock нет необходимости в сторонних менеджерах буфера обмена, в cairo-dock есть свой прекрасный менеджер буфера

clipper.

**compton -** композитный менеджер, форк xcompmgr с расширенными возможностями и без его глюков. Сорцы здесь — github.com/chjj/compton. Конфигурировать ничего не надо, просто создать пакет скриптом. Заодно можно перекомпилировать с заменой флагов компиляции и линковки и сам xcompmgr, если почему то не понравился compton.

**cairo-dock, cairo-dock-plugins -** док в стиле макоси.

Есть на slackbuilds.org, но последние версии разжиревшие сверх меры в плане ресурсоемкости, поэтому лучше поставить более старую версию.

Слакбилд cairo-dock - repository.slacky.eu/slackware-13.1/desktop/cairo-dock/2.1.3_9/src (сорцы сам подтянет).

Слакбилд cairo-dock-plugin- repository.slacky.eu/slackware-13.1/desktop/cairo-dock-plugins/2.1.3_9_2/src (сорцы сам подтянет).

Или берем здесь - www.wuala.com/SergMarkov19/Slackbuilds/cairo-dock-2.1.3-9

Правим слакбилд cairo-dock-plugin, так как mail plugin с новыми версиями gcc не собирается, поэтому заменяем

```
./configure \
  --prefix=/usr \
  --libdir=/usr/lib${LIBDIRSUFFIX} \
  --sysconfdir=/etc \
  --localstatedir=/var \
  --mandir=/usr/man \
  --docdir=/usr/doc/$PKGNAME-$VERSION \
  --disable-static \
  --program-prefix= \
  --program-suffix= \
  --enable-mail=no \
  --build=$CHOST-slackware-linux
```

на

```
./configure \
  --prefix=/usr \
  --libdir=/usr/lib${LIBDIRSUFFIX} \
  --sysconfdir=/etc \
  --localstatedir=/var \
  --mandir=/usr/man \
  --docdir=/usr/doc/$PKGNAME-$VERSION \
  --disable-static \
  --program-prefix= \
  --program-suffix= \
  --enable-mail=no \
  --build=$CHOST-slackware-linux
```

Любителям минимализма может понравится маленькая выплывающая сверху панель для запуска программ yeahlaunch — www.bstern.org/yeahlaunch.

Есть еще одна кандидатура на роль дока - Avant Window Navigator,но он требует композитного менеджера , как минимум xcompmgr, что не всегда приемлемо, к тому же именно с xcompmgr awn работает абы как.

Далее ставим проги со slackbuilds.org (читаем там же о зависимостях и ставим их). Естественно правим слакбилд как выше сказано и собираем с последней версией прог, сорцы которых берем с их сайта.

**gdmap** - графическая карта дисков
**gprename** - переименование файлов
Нужны пакеты perl-extutils-depends, perl-extutils-pkgconfig, perl-glib, perl-test-number-delta, perl-cairo,perl-pango, perl-gtk2, perl-libintl, locale-gettext со slackbuilds.org
**wxhexeditor** - HEX редактор (даже если не ставить его самого, то для некоторых программ далее понадобится wxPython, который входит в его зависимости). В слакбилде не меняем опции компиляции.
**isomaster** - для работы с образами ISO
**meld** - сравнение файлов и папок
**bleachbit** - уборка мусора
**gsmartcontrol** - контроль SMART
**gtk-chtheme** - переключалка тем GTK
**htop** - менеджер процессов
**wmctrl** - утилита управления параметрами окон
**numlockx** - включение NumLock
**xonclock** - часики со скинами

Архиваторы, синхронизация и резервное копирование
**p7z** — архиватор
**unrar** — архиватор

Как настроить корректное отображение кириллических имен файлов в архивах zip описано здесь — www.opennet.ru/tips/info/2494.shtml. Но самый простой способ он как всегда самый лучший :-), благо таких архивов очень мало , просто ставим под wine любой виндовый архиватор – 7z, peazip, winrar.

**file-roller** — гуй к архиваторам
**luckybackup** — бэкапы файлов и директорий
**davfs2** – доступ по webdav к облакам
**fsarchiver**— бэкапы директорий и разделов (не применяем для и на ntfs)
Для fsarchiver есть хороший гуй, там же есть и LiveCD с ним.
**qt4-fsarchiver** — sourceforge.net/projects/qt4-fsarchiver

```
qmake
make all
```

Получившийся бинарник в /usr/local/bin и прописываем его в /etc/sudoers.
Есть слакбилд на www.wuala.com/SergMarkov19/Slackbuilds

**Super Flexible File Synchronizer** - программа для облачной синхронизации и резервного копирования, поддерживающая FTP, SSH, WebDAV, Amazon S3 Google Docs. - www.superflexible.com/linux.htm

**dropbox client** — синхронизация с dropbox
Бинарник - www.getdropbox.com/download?plat=lnx.x86

**wuala client** — синхронизация и резервное копирование wuala.com
wuala.com/en/download/linux.

Для переименования файлов удобно использовать скрипт **vilm**, который можно повесить на кнопку worker. Он позволяют использовать для переименования файлов текстовый редактор, во многих случаях это удобнее чем использовать специальную утилиту. Скрипт кладем куда нибудь в path, в /usr/local/bin к примеру.

**vilm** — redchamp.net/vimv
Для использование juffed вместо vi правим скрипт, заменяем

```
parser.add_option('-e',
                  '--editor',
                  default='vi',
                  help = 'Use an alternative editor (e.g., pico).')
```

на

```
 parser.add_option('-e',
                  '--editor',
                  default='juffed',
                  help = 'Use an alternative editor (e.g., pico).')
```

**hddtemp** - температура винтов
Компиляция hddtemp пример того, как можно создать пакет программы, которая обычным способом не компилируется. Скачиваем сорцы hddtemp с патчами для последней версии убунты, в которой его еще можно найти, с packages.ubuntu.com. Копируем все патчи в папку с распакованными сорцами hddtemp (в разные папках патчей могут быть патчи с одинаковыми названиями, естественно применяем их по очереди и все). Применяем все патчи, создаем скриптик в папке с распакованными сорцами, делаем исполняемым и запускаем в терминале

```
#!/bin/bash
clear
for d in ls /mnt/sda9/Slackware/Compile/hddtemp-0.3-beta15/*; do
patch -p1 -i `basename "$d"`;
echo "----------------------"
done
exit 0
```

Конфигурация

```
./configure \
--build=i686-pc-linux-gnu \
```

```
--host=i686-pc-linux-gnu \
--prefix=/usr \
--libdir=/usr/lib \
--localstatedir=/var \
--sysconfdir=/etc \
--mandir=/usr/man \
--with-db-path=/etc/hddtemp.db \
CFLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
-pipe  -mmmx -m3dnow -falign-jumps=1 -falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
-pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1" \
LDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

Это приведено лишь в качестве примера :-), на самом деле он есть на slackbuilds.org

**wine -** неэмулятор виндов.
Сначала ставим зависимости

**webcore-fonts** - шрифты MS со slackbuilds.org

Перед установкой webcore-fonts переносим куда-нибудь в другое место файл /etc/fonts/conf.d/60-liberation.conf (еще лучше заархивировать весь/etc/fonts/conf.d) и после установки возвращаем на место. Если будут какие то проблемы с замещением шрифтами от MS своих шрифтов, то устанавливаем в каждой секции каждого файла в /etc/fonts/conf.d свои шрифты выше шрифтов от MS.
**winetricks** - установка библиотек и разных причиндал виндов для wine со slackbuilds.org
**cabextract** со slackbuilds.org
**fontforge** - редактор шрифтов со slackbuilds.org
Теперь ставим сам **wine** со slackbuilds.org.

**font-manager** - менеджер шрифтов, которым очень удобно настраивать параметры шрифтов, такие как сглаживание и прочие, вместо копания в fonts.conf, позволяет также включать и выключать шрифты или группы шрифтов.
Сорцы - code.google.com/p/font-manager

```
./configure \
--build=i686-pc-linux-gnu \
--host=i686-pc-linux-gnu \
--prefix=/usr \
--libdir=/usr/lib \
--localstatedir=/var \
--sysconfdir=/etc \
--mandir=/usr/man \
--enable-nls \
CFLAGS="-O3 -march=native -mtune=native -fomit-frame-pointer \
-pipe -mmmx -m3dnow -falign-jumps=1 -falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1" \
```

```
LDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

Настраиваем шрифты через font-manager, сглаживание и тому подобные феньки и обязательно отключаем arial.

Возвращаем на место /etc/fonts/conf.d/60-liberation.conf.

**freetype** - библиотека шрифтов.

Поскольку fontforge собирается используя свой freetype без патчей и с далеко не лучшей конфигурацией по умолчанию, пересобираем freetype из слакбилда с установочного DVD.

**Улучшение шрифтов**

Немного о самой freetype и вообще об улучшении шрифтов. Есть несколько способов значительно улучшить шрифты. Восприятие шрифтов сугубо субъективно, кому нравится одно, кому другое, поэтому если не подходит какой то из способов можно попробовать другой, а в рамках самого выбранного способа побаловаться настройками font-manager. Как правило нужный вид шрифтов таким методом всегда можно подобрать, затратив не такое уж и большое время.

Важно — перед тем как что то делать с настройками шрифтов, архивируйте /var/cache/fontconfig,/etc/fonts,~/.fonts,~/.fontconfig,~/.fonts.conf,~/.config/font-manager,~/.config/Trolltech.conf,~/.config/fontgroups.xml, чтобы потом не плакать «пропало все». Архивировать их надо всегда, когда наковыряли настройки шрифтов так, что они превратились в нечто совсем неудобоваримое, или использовали для настройки внешнего вида прог KDE systemsettings, который успешно херачит ваши настройки шрифтов, даже если вы их в systemsettings вообще не трогали.

- Первый способ - использовать возможности конфигурации самой freetype без дополнительный патчей, включить bytecode interpreter (если он выключен) и включить масштабирование и сглаживание от Apple.

Распаковываем архив с сорцами, идущими в комплекте со слакбилдом freetype, правим /freetype-***/include/freetype/config/ftoption.h
раскомментируем, чтобы было вот так

```
#define TT_CONFIG_OPTION_BYTECODE_INTERPRETER
```

и закомментируем, чтобы было вот так

```
/* #define TT_CONFIG_OPTION_UNPATENTED_HINTING */
```

Переопределим чтобы было вот так

```
#define TT_CONFIG_OPTION_COMPONENT_OFFSET_SCALED
```

Естественно если не нравятся феньки от Apple то последнее можно убрать.

Для улучшения отображения на LCD мониторах включаем в нем субпиксельный рендеринг, для чего раскомментируем в слакбилде строчку

```
zcat $CWD/freetype.subpixel.rendering.diff.gz | patch -p1 --verbose || exit 1
```

раскомментируем также

```
zcat $CWD/freetype.illadvisederror.diff.gz | patch -p1 --verbose || exit 1
```

Запаковываем сорцы в архив и подсовываем архив слакбилду freetype. Если не нравится версия freetype заменяем ее какой нибудь (лучше естественно последней версией с сайта freetype) и также правим ее.

Включаем сам субпиксельный рендеринг - из /etc/fonts/conf.avail делаем симлинк в /etc/fonts/conf.d одного из файлов 10-sub-pixel-*.conf. Какого именно рендеринг - какой больше понравится на глаз :-), делаем скриншоты для каждого файла и сравниваем. Скорее всего это будет vrgb рендеринг.

В принципе самый простой способ как всегда самый лучший :-), и кому нравятся четкие, контрастные и сглаженные шрифты лучше всего ограничиться включением bytecode interpreter и сглаживание от Apple, со включенными патчами в слакбилде, даже без симлинков sub-pixel-* в /etc/fonts/conf.d, и включить в шрифтах полный (full) хинтинг (hinting) и сглаживания (antialisang) для других шрифтов. Впрочем кому нравятся чуть сглаженные шрифты, можно их в font manager и не включать.

Чисто личные предпочтения - шрифты по первому способу компиляции freetype в разделе 'Улучшение шрифтов', сглаживание и полное уточнение для всех шрифтов, нет субпиксельного сглаживания (линков **-sub-pixel в /etc/fonts/conf.d), шрифт интерфейса, текстовых редакторов и браузера Liberation Sans, шрифт читалки handbookpdcyr, serif для офисов AdonisC, шрифт терминалов DejaVu Sans Mono и Terminus. Почему именно так, ответ прост и правдив - они нравятся :-). Оставлены нужные для wine и виндовых доков Arial(см. ниже об алиасах), Times New Roman и MS Sans Serif, остальные шрифты за ненадобностью выключены в font-manager.

Затем чистая мистика :-) для случая личных предпочтений, почему так происходит внятного объяснения нет, возможно на это влияет наличие папки ~/.compose-cache, поэтому ее лучше создать заранее. Но шрифты становятся еще лучше. Делается так:

- Включается в font-manager Arial с полным уточнением и сглаживанием
- Убирается симлинк /etc/fonts/conf.d/60-liberation.conf
- В default.theme текушей темы IceWM заменяется все *FontName*
  для примера одной такой замены, на
  TitleFontNameXft = Arial:size=14
- Перезапускается тема IceWM, в его меню заново выбрать ту же тему
- В default.theme текущей темы IceWM заменяется все *FontName* на
  для примера одной такой замены
  TitleFontNameXft = Liberation Sans:size=14
- Перезапускается тема IceWM, в его меню заново выбрать ту же тему
- Возвращается на место симлинк /etc/fonts/conf.d/60-liberation.conf
- Перезапускается тема IceWM, в его меню заново выбрать ту же тему
- В default.theme текушей темы IceWM заменяется все *FontName*
  для примера одной такой замены, на
  TitleFontNameXft = sans-serif:size=14
- Перезапускается тема IceWM, в его меню заново выбрать ту же тему

и шрифты становятся еще лучше. Почему они от этого становятся лучше, ну так это и есть мистика :-)

Далее, в случае установки webcorefonts и использования Liberation Sans для интерфейса и всего другого в качестве sans-serif, надо включить Liberation Sans в список алиасов в /etc/fonts/conf.avail/60-latin.conf, привести секцию

<family>sans-serif</family> к следующему виду

```
<family>sans-serif</family>
<prefer>
        <family>Liberation Sans</family>
        <family>DejaVu Sans</family>
        <family>Bitstream Vera Sans</family>
        <family>Verdana</family>
        <family>Arial</family>
        <family>Albany AMT</family>
        <family>Luxi Sans</family>
        <family>Nimbus Sans L</family>
        <family>Helvetica</family>
        <family>Lucida Sans Unicode</family>
        <family>BPG Glaho International</family> <!-- lat,cyr,arab,geor -->
        <family>Tahoma</family> <!-- lat,cyr,greek,heb,arab,thai -->
</prefer>
</alias>
```

Это сделано для того, чтобы можно было использовать включенный в font-manager Arial, без того чтобы он подменял Liberation Sans.

- Второй способ - шрифты а-ля ubuntu
На freetype, fontconfig, libXft, cairo накладываются патчи от Дугана

duganchen.ca/writings/slackware/fonts, раздел LCD Filter Rendering, или
github.com/duganchen/slackware-lcdfilter

Можно посмотреть и здесь

http://gitorious.org/lcd-filtering
https://raw.github.com/duganchen/dotfiles/master/.fonts.conf

или как вариант брать патчи отсюда

https://launchpad.net/ubuntu/oneiric/+source/freetype
https://launchpad.net/ubuntu/oneiric/+source/fontconfig
https://launchpad.net/ubuntu/oneiric/+source/cairo
https://launchpad.net/ubuntu/oneiric/+source/xft

Включаем сам субпиксельный рендеринг - из /etc/fonts/conf.avail делаем симлинк в /etc/fonts/conf.d одного из файлов 10-sub-pixel-*.conf.

- Третий способ - шрифты а-ля Microsoft clear type
На freetype, libXft, cairo накладываются патчи отсюда (в комплекте сами патчи и слакбилд)

https://sites.google.com/site/mostlyslack/cleartype

Там же примеры fonts.conf и скрины до и после

Включаем сам субпиксельный рендеринг - из /etc/fonts/conf.avail делаем симлинк в /etc/fonts/conf.d одного из файлов 10-sub-pixel-*.conf. Убираем линк /etc/fonts/conf.d/60-liberation.conf и выбираем Arial как шрифт интерфейса и всего другого.

Есть также набор патчей от www.infinality.net/blog, включающий субпиксельное сглаживание и пример fonts.conf.

Есть множество других патчей для улучшения шрифтов, это только основные их виды. Естественно даже их можно комбинировать и получить что то, может и лучшее а может и вырвиглазное :-)

Пересобрать freetype со включенным bytecode interpreter и subpixel rendering лучше в любом случае, так как шрифты с ними смотрятся на порядок красивее.

**glogg** — просмотр и удобный поиск в больших (более 10 Мб) файлов
http://glogg.bonnefon.org/index.html
Конфигурация стандартная для qmake.

Также берем готовые пакеты со slackfind.net

**xvidcap** - запись скринкастов (требует lame со slackbuilds.org)
**hardinfo** - информация о системе (требует libsoup со slackbuilds.org)
**aspell-ru -** русский словарь

**Бенчмарки**
www.corecrowd.com/qtperf.tar.bz2 - бенчмарк QT
code.google.com/p/qtperf - другой бенчмарк QT
gtkperf.sourceforge.net - бенчмарк GTK

## 2 - Интернет

**google chrome** — браузер.
Устанавливаем из dvd://extra пакет google-chrome-pam-solibs (в дальнейших версия слаки это может измениться).

Скачиваем релиз хрома - пакет deb, кладем в папку с ним google-chrome.SlackBuild из dvd://extra и запускаем слакбилд, далее устанавливаем получившийся пакет.

Если не нравятся продукция империи добра или хочется более новую версию, то на slacky.eu есть chromium.

**firefox** - если не нравится хром или уже установленная версия FF и хочется поновее, скачиваем архив или с релизом или с ночнушками с ftp.mozilla.org/pub/firefox, распаковываем архив в /opt и делаем симлинк с firefox в /usr/local/bin.

Если не нужен старый firefox из самой слаки, то удаляем его и копируем содержимое папки firefox скачанного архива в /usr/lib/firefox-** , далее делаем симлинк из /usr/lib/firefox-*/firefox в /usr/bin/firefox.

**dillo** - браузер для быстрого просмотра локальных страничек, он очень быстр, запускается практически мгновенно и удобен именно для локального просмотра, когда надо быстренько глянуть какую то сохраненную страничку. Лучше брать версии 2.***, поскольку версии 3.**, прибавив в возможностях просмотра инета, ничего нового по сути не дали для просмотра локальных страничек, но заметно утяжелили и сам браузер и время его запуска.

Нужен тулкит fltk2 со slackbuilds.org. Чтобы не загромождать систему лишним тулкитом, под которым практически нет вменяемых программ кроме dillo и быстрого, но малофункционального и глюкавого файлового менеджера xfe, сделаем статическую линковку dillo с fltk2. Для этого правим слакбилд fltk2, заменяем

```
./configure \
  --prefix=/usr \
  --libdir=/usr/lib${LIBDIRSUFFIX} \
  --mandir=/usr/man \
  --enable-shared
```

на

```
./configure \
  --prefix=/usr \
  --libdir=/usr/lib${LIBDIRSUFFIX} \
  --mandir=/usr/man
```

Далее собираем сам dillo со slackbuilds.org и удаляем после сборки fltk2, он уже не нужен.

**claws-mail** - почтовый клиент.
Сначала устанавливаем нужные пакеты
icu4c, libetpan, libsoup - со slackbuilds.org
compface - со slackfind.net
webkitgtk - со connie.slackware.com/~alien/slackbuilds
Ставим сам claws-mail со slackbuilds.org

Ставим плагины к нему claws-mail-extra-plugins со slackbuilds.org, плагинов полно, в том числе и ненужные, поэтому правим слакбилд и отключаем ненужные. Заменяем в слакбилде

```
PLUGIN_LIST="acpi_notifier address_keeper archive att_remover attachwarner \
  bsfilter_plugin clamd fancy fetchinfo-plugin geolocation_plugin \
  gtkhtml2_viewer mailmbox newmail notification_plugin perl_plugin \
  python_plugin rssyl spam_report tnef_parse vcalendar"
```

на

```
PLUGIN_LIST="address_keeper archive attachwarner
  fetchinfo-plugin fancy \
  mailmbox \
  tnef_parse"
```

Есть и готовые пакеты на rlworkman.net/pkgs/13.37/i486.

**logjam** - клиент LiveJournal.
Сорцы - andy-shev.github.com/LogJam
Конфигурация по шаблону
В нем есть возможность предпросмотра через gtkhtml, но оно тянет полгнома.

**filezilla** - FTP клиент со slackbuilds.org.
Правим слакбилд
LDFLAGS="-ldl -Wl,-O1 -Wl,--as-needed" \
Если не ставили wxhexeditor то нужен wxPython со slackbuilds.org.

**qbittorrent** - торрент клиент а-ля utorrent
Сорцы — www.qbittorrent.org. Нужна либа libtorrent-rasterbar со slackbuilds.org. Есть также слакбилд на slackbuilds.org.
Для использования встроенной базы IP адресов для отображения флажков стран пиров скачиваем саму базу с
geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz
и кладем распакованный файл с ней в ../src/geoip
Конфигурация

```
./configure --prefix=/usr --with-geoip-database-embedded
```

Для сборки пакета необходимо использовать не makepkg a slacktrack.
В инете есть упоминания что qbittorrent тормозит на стопицтот торрентах, ни подтвердить ни опровергнуть не могу, поскольку редко когда бывает более 15 торрентов, из которых не более 5 активных. Но если нужна работа с большим, более 100, количеством торрентов, то может быть полезен

**transmission** - торрент клиент со slackbuilds.org.
Нужна либа libevent со slackbuilds.org.
Удаляем из ./configure

```
--disable-static
```

и убираем создание корявого qt гуя - удаляем из слакбилда

```
cd $TMP/$PRGNAM-$VERSION/qt
  qmake \
    QMAKE_CXXFLAGS+="$SLKCFLAGS" \
    QMAKE_CFLAGS+="$SLKCFLAGS" \
    qtr.pro
  sed -i -e 's% -g % %g' Makefile
  make
  INSTALL_ROOT=$PKG/usr make install
cd -
```

**gmediafinder** — поиск, просмотр и загрузка клипов с множества видеохостингов, таких как youtube.com.
Сорцы - github.com/smolleyes/gmediafinder
Нужны пакеты configobj, gdata, mechanize со slackbuilds.org. Создание пакета

стандартное для питона. Есть слакбилд на www.wuala.com/SergMarkov19/Slackbuilds. Также может быть полезен **minitube** или **miro** со slackbuilds.org.

**licq** — ICQ клиент со slackbuilds.org. В последних версиях пофиксены все баги с кириллицей и самый удобный асечный клиент вполне готов к работе с ней. Нужны либы cdk, xosd, libaosd со slackbuilds.org. Если в очередной раз изменится протокол аськи и это изменение еще не будет внесено в основную версию, то собираем из git. Клонируем git

```
git clone git://github.com/licq-im/licq.git
cd build-all
```

Далее стандартная конфигурация cmake.

Кому нравятся комбайны есть **qutim —** qutim.org.
Для него нужны qca2-cyrus-sasl и libqxt со slackfind.net, и libjreen со qutim.org/jreen. Конфигурация стандартная для cmake.

**vacuum** - jabber клиент.
Сорцы - code.google.com/p/vacuum-im/downloads/list
Конфигурация стандартная для cmake

**qtwitter** - twitter клиент.
Со slackbuilds.org. Нужна либа qoauth со slackfind.net.

Есть также хороший клиент hotot с pkgs.org и turpial (сорцы – files.turpial.org.ve/sources/stable, в зависимостях – Babel, pygtkspell, notify-python с pkgs.org). Оба последних хороши по возможностям, но несколько тормозные.

**fatrat** - менеджер загрузок, также может быть использован как торрент клиент с хорошей функциональностью, которой вполне хватает для большинства задач. Для торрентов использует ту же либу libtorrent-rasterbar что и qbittorrent.
Сорцы — fatrat.dolezel.info
Конфигурация

```
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/usr -DWITH_CURL=ON \
-DWITH_BITTORRENT=ON -DWITH_NLS=ON
```

**webhttrack**- загрузка сайтов.
Сорцы - www.httrack.com
Конфигурация по шаблону.

**aMule** - P2P клиент со slackbuilds.org.
Нужны либы cryptopp(читаем замечания на страница слакбилда aMule, libupnp, geoip со slackbuilds.org.
Правится slackbuild

```
./configure \
--prefix=/usr \
--libdir=/usr/lib${LIBDIRSUFFIX} \
--sysconfdir=/etc \
```

```
--localstatedir=/var \
--mandir=/usr/man \
--with-wx-config=/usr/bin/wx-config \
--disable-debug \
--build=$ARCH-slackware-linux \
--enable-amule-daemon \
--enable-amulecmd \
--enable-amule-gui \
--enable-cas \
--enable-wxcas \
--enable-alc \
--enable-alcc \
--enable-geoip \
--disable-debug \
--with-denoise-level=3
```

**gtorrentviewer** - просмоторщик торрент-файлов, бывает удобен когда накопилось много торрент-файлов с непонятными названиями.
Сорцы - gtorrentviewer.sourceforge.net
Конфигурация по шаблону.

**w3m** - консольный браузер, тоже иногда может быть полезен, со slackbuilds.org.
Нужен пакет gc со slackbuilds.org.

### 3 - Графика

С графикой более или менее просто, на DVD есть gimp для редактирования растровых изображений и geeqie для просмотра множества форматов, есть xpaint -snapshot как скриншотилка, со slacky.eu ставится inkscape как векторный редактор (если не ставили ранее regexxer то придется сейчас ставить нужные для него либы). Так что остается вообще то немного маленьких программ.

**screengrab -** скриншотилка с возможностью выгрузки на imageshack.us со slackbuilds.org.

Есть также скриншотилка **Jshot** — jshot.info с возможностью редактирования и выгрузки на imageshack.us (ставить его надо в /home иначе не подхватывает свои плагины).

Широко известный в узких кругах shutter тянет за собой все либы гнома.

**gcolor2** - подбор цвета со slackbuilds.org.

**xcolmix -** еще одна утилита подборки цвета, позволяет оценить цвет текста на цвете фона.
Сорцы - ftp.de.debian.org/debian/pool/main/x/xcolmix/xcolmix_1.07.orig.tar.gz
Нужна тулкит xforms со slackbuilds.org.

Поскольку под xforms сейчас вообще нет программ, кроме разве что xcolmix, то делаем статическую линковку xcolmix с xforms, чтобы после компиляции xcolmix можно было удалить xforms, Для этого приводим секцию configure в слакбилде xforms к виду

```
./configure \
  --prefix=/usr \
  --libdir=/usr/lib${LIBDIRSUFFIX} \
  --docdir=/usr/doc/$PRGNAM-$VERSION \
  --datadir=/usr/share/$PRGNAM-$VERSION \
  --mandir=/usr/man \
  --infodir=/usr/info \
  --enable-demos \
  --enable-docs \
  --enable-shared=no \
  --build=$ARCH-slackware-linux
```

комментируем строчку чтобы было вот так

```
# cat $CWD/xforms.pdf > $PKG/usr/doc/$PRGNAM-$VERSION/xforms.pdf
```

и заменяем

```
DEMOFILES="01Readme *.h *.c *.c.old *.xbm *.xpm *.menu .libs/*"
```

на

```
#DEMOFILES="01Readme *.h *.c *.c.old *.xbm *.xpm *.menu"
```

Правим сорцы xcolmix, в src/design.c заменяем

```
#include <X11/forms.h>
```

на

```
#include <forms.h>
```

аналогичную замену проводим в src/xcolmix.h
Компиляция - make final

получившийся файл в /usr/local/bin.
Теперь удаляем xforms, он уже не нужен.

**gammapage** - настройка gamma и цветовая калибровка, ей можно регулировать и яркость
Готовый пакет - ftp://ftp.yandex.ru/altlinux/Sisyphus/files/SRPMS/gammapage-0.5.1-alt1.src.rpm
(это простой скрипт на питоне, вытаскивается из архива при помощи file-roller).

**inkscapelite -** замена inkscape для простых операций (коих большинство, и для которых нет никакой необходимости запускать такое многотонное угробище как сам inkscape). С его помощью можно быстренько подправить, создать что то ненавороченное, конвертнуть в svg.
Сорцы - www.murga-linux.com/puppy/viewtopic.php?p=369333#369333

```
./configure \
--build=i686-pc-linux-gnu \
```

```
--host=i686-pc-linux-gnu \
--prefix=/usr \
--libdir=/usr/lib \
--localstatedir=/var \
--sysconfdir=/etc \
--mandir=/usr/man \
--with-pic \
--with-gnome-print=no \
--with-xft \
CFLAGS="-O3 -march=native -mtune=native -pipe \
-fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native -pipe \
-fomit-frame-pointer -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1"
```

Перед установкой пакета переименовываем входящий в него inkscape в inkscape-lite, чтобы не было путаницы с самим inkscape, распаковываем куда нибудь пакет, переименовываем inkscape в inkscape-lite и запаковываем пакет makepkg по аналогии с созданием пакета при компиляции.

**Редакторы диаграмм**

Нормальных редакторов диаграмм по сути два - **dia** и **yEd**. Первый берется со slackbuilds.org, второй (архив с jar) со www.yworks.com/en/products_yed_applicationfeatures.html.
Запуск второго стандартный - java -Xmx256M -jar yed.jar.

**geeqie** - просмотрщик изображений.

Для ускорения работы и использования libjpeg-turbo в geeqie лучше перекомпилировать его из сорцов с DVD с заменой в слакбилде флагов компиляции на свои.

Если не нужна поддержка стопицтот форматов, то лучше использовать предка geeqie — **gqview**, сорцы — gqview.sourceforge.net. Он быстрее и запускается и работает. Конфигурация стандартная.

Впрочем, как это не печально для любителей opensource, но лучшая на сегодня программа для просмотра изображений это проприетарная XnViewMP — newsgroup.xnview.com/viewtopic.php?f=60&t=24056. Богатейшие возможности, прекрасное удобство и при все при этом высочайшая скорость работы, выше чем в опенсурсных аналогах. Широкие возможности коррекции изображений прямо в программе, выгрузка на фотохостинги, система рейтингования, подключение внешних прог, и не требует ничего для работы кроме phonon. Установка крайне проста — разархивируем в какую то папку, хоть в opt хоть в папку для программ в home, корректируем xnview.sh, вставляя сразу после #!/bin/sh

```
cd /папка с XnViewMP
```

делаем симлинк из xnview.sh в /usr/local/bin и прописываем в файловые зависимости worker.

Есть еще естественно и picasa, но это редкий тормоз. Продолжая тему нельзя не упомянуть и Corel AfterShot Pro for Linux, программу обработки изображе-

ний от Corel, как и все ее продукты она прекрасна, также естественна что она платная, но наверное все знают как приобретать лицензию на продукты Corel .. :-)

## 4 - Мультимедиа

Сначала ставятся либы - lame, aften, x264, mac, xvidcore, xvid4conf, cfourcc, libdv, libdvbpsi, libdvdcss, libdvdnav, libmpeg2, twolame, mppenc, faac, faad2, libmp4v2, rtmpdump, a52dec, libdca, libmms, libvpx, live555, mjpegtools, oggvideotools, speex, OpenAL, freealut, libvdpau, libass, mplayer-codecs, libiconv со slackbuilds.org. Либ много, но все они маленькие и компилируются за полминуты.

В слакбилде libvpx комментируем, чтобы было вот так

```
#cp -a AUTHORS CHANGELOG LICENSE PATENTS README build-tmp/docs/html \
#  $PKG/usr/doc/$PRGNAM-$VERSION
```

**ffmpeg** со slackbuilds.org

В слакбилде заменяем для faac, faad, rtmp, speex, vpx, xvid "no" на "yes"и комментируем строчку

```
# cp -a doc/*.html $PKG/usr/doc/$PRGNAM-$VERSION
```

**mplayer2** - форк mplayer с расширенными возможностями, в остальном он полностью совместим с классическим mplayer.
Сорцы самого mplayer2 берем со www.mplayer2.org, клонируем git

```
git clone git://git.mplayer2.org/mplayer2-build.git
```

Перед компиляцией правим ~/.bashrc, временно заменяем

```
export CFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1"
```

на

```
export CFLAGS="-O2 -march=i486 -mtune=i686"
```

и комментируем, чтобы было вот так

```
#export LDFLAGS="-Wl,-O1 -Wl,--as-needed"
```

перезагружаем иксы для вступления изменений в силу.

Далее make и получившийся бинарник mplayer2-build-*/mplayer/mplayer в /usr/local/bin. После компиляции mplayer2 производим обратную замену в ~/.bashrc.

**mplayer** — если по каким то причинам нужен классический mplayer то получаем сорцы из svn, так как разные версии mplayer используют разные версии ffmpeg и совпадают они только в svn версии, ffmped загружается из git по запросу при кон-

фигурации

```
svn checkout svn://svn.mplayerhq.hu/mplayer/trunk mplayer
cd mplayer
svn update
```

Конфигурация, в ней нет mencoder, tv и radio, если есть соответствующее железо (тюнеры и.т.п), то включаем их

```
./configure \
--prefix=/usr \
--mandir=/usr/man \
--confdir=/etc/mplayer \
--enable-menu \
--disable-mencoder \
--disable-tv \
--language-man=ru \
--language-msg=ru \
--language-doc=ru \
--enable-dynamic-plugins \
--disable-arts \
--enable-runtime-cpudetection \
--codecsdir=/usr/lib/codecs \
--language="en ru"
```

Перед установкой пакета переименовываем mplayer в mplayer-classic, чтобы не было путаницы с mplayer2

Для продолжения прослушивания/просмотра аудио и видео файлов с той позиции, на которой просмотр/прослушивание завершился в прошлый раз при закрытии Mplayer'a есть обертка к нему, mplayer.ext — sourceforge.net/projects/mplayerext

**umplayer** - фронтенд к mplayer со многими дополнительными возможностями, такими как просмотр и поиск клипов с youtube.com, прослушивание интернет радио с менеджером станций. Есть на slackbuilds.org.

Можно поставить smplayer со slackbuilds.org, но в нем неудобно организован доступ к youtube и showcast.

Во всех форках smplayer - umplayer, smplayer2, и в нем самом, есть некоторое неудобство при регулировке параметров с клавиатуры, слишком велик шаг изменения параметров и поэтому трудно подстроить их оптимальные значения. Проблема решается правкой сорцов. В файле ../src/core.cpp есть функции void Core::incBrightness(), void Core::decBrightness(), void Core::incContrast(), void Core::decContrast(), void Core::incGamma(), void Core::decGamma(), void Core::incHue(), void Core::decHue(), void Core::incSaturation(), void Core::decSaturation(), в которых надо заменить set*(mset.* +(-) 4); на set*(mset.* +(-) 1); и шаг регулировки станет минимальным.

Кому не нравится mplayer с форками всегда есть **vlc** — taper.alienbase.nl/mirrors/people/alien/restricted_slackbuilds/vlc, естественно надо брать версию без патентных ограничений.

Есть симпатичный плеер xt7-player — xt7-player.sourceforge.net/xt7forum. В некотором роде подтверждение того вполне понятного факта, что нет плохих языков есть криворукие программеры:-) Плеер написан на gambas, но по количеству настроек, продуманности их использования, по дизайну, этот гуй великолепен, и превосходит такие известные гуи как smplayer или umplayer. В нем есть такая приятная плюшка как перехват флеша плеером, причем с возможностью автоматического сохранения (естественно сам плеер для такого перехвата должен быть запущен, никакого демона нет). Конечно есть масса скриптов для такого перехвата, но скрипта с сохранением флеша пока не видел.

В нем есть только один, но вполне понятный глюк, не работает увеличение приоритета mplayer через renice, встроенное в настройки. Выход простейший, или ставим SUID на копию renice (типа renblabla) или прописываем копию renice в sudoers. И в том и в другом случае правим соответствующим образом в xt7-player-3.1/2/xt7-player/.src/Classi/MplayerClass.class жирное выделение

```
Public Sub Renice(nice As Integer)
  Dim out As String
  Debug "nice=" & nice
  Debug
  RenicerProcess = Shell "sudo renblabla " & nice & " -p " & Application.id  'renice the whole application to 'nice'
  RenicerProcess = Shell "sudo renblabla " & (nice - 1) & " -p " & playerprocess.id  'renice the whole application to 'nice'-1
End
```

Для компилирования нужны пакеты gambas3 и gambas3-runtime со slacky.eu. Конфигурация стандартная. После компилирования gambas3 (не -runtime) можно удалить.

**gnome-mplayer -** фронтенд к mplayer, необходим только для проигрывания флеша с youtube.com через расширение firefox flashvideoreplacer, со slackbuilds.org. Впрочем можно с успехом использовать в нем и xine.

**xine-plugin** - плагин к браузерам на основе gecko для проигрывания потокового видео через xine, сорцы www.xine-project.org/releases. Конфигурация стандартная. Аналогичный плагин к mplayer обладает способностью работать абы как, к тому же зависит от версии браузера,и если захочется сменить фокс на новый, он может отказаться делать ку :-)

**fbreader** - читалка книг, со slackbuilds.org.
Нужна либа liblinebreak со slackbuilds.org.
В слакбилде fbreader заменяем

```
UI=${UI:-qt4}
```

на

```
UI=${UI:-gtk}
```

и заменяем

```
patch -p1 < $CWD/fbreader_gcc45x.patch
```

на

```
patch -p1 < $CWD/fbreader_gcc45x.patch
LDFLAGS="-Wl,-O1 -Wl,--as-needed" \
```

Есть еще одна хорошая читалка книг, CoolReader, со slackbuilds.org

**MyRuLib** — каталогизатор онлайн библиотек, позволяющий каталогизировать по темам, авторам и сериям с использованием собственной обновляемой базы книги с флибусты, либрусека и некоторых других библиотек, с возможностью их скачивания и чтения, со slackbuilds.org.

Немного о выборе аудиоплеера. Если выбор видеоплееров по сути ограничен mplayer, vlc и xine с различными их фронтендами и форками, то аудиоплееров наплодили несметное количество и выбор между ними сделать довольно таки непросто. Поэтому сначала определяемся с критериями такого выбора, которые естественно у каждого свои, поэтому далее сугубо субъективные рассуждения :-)

Что нужно от аудиоплеера, естественно в первую очередь хороший звук. Хороший звук требует как хорошего кода самого плеера так и хорошего эквалайзера, в силу объективной неравномерности АЧХ всего процесса восприятия звука - от записи до прослушивания через среднего качества колонки в комнате с коврами и трухлявым паркетом :-) Далее, если есть большое количество музыки то в плеере нужна база данных, которая поддерживает все форматы которые есть в наличии. Ну и естественно всякие дополнительные плюшки типа поддержки интернет-радио, last.fm, поиска обложек и текстов, которые в принципе совершенно необязательны, поскольку аудиоплеер обязан иметь только две основные характеристики, хороший звук и хороший базу с поиском в ней. Исходя из всего этого как аудиоплеер был выбран guayadeque, с прекрасным, без малейшей глухоты, звуком, отличным эквалайзером, с sqlite базой данных, понимающей все форматы, в том числе и ape и возможностью выборки из этой базы по разным критериям и их комбинациям. Плюс все дополнительные плюшки. Он не очень красив, но на аудиоплеер как правило не глядят а слушают через него музыку, так что это не самый большой недостаток. Guayadeque требует wxPython, так что если раньше его не поставили, то теперь уж точно понадобится :-)

**guayadeque** — аудиоплеер.
Сорцы — sourceforge.net/projects/guayadeque. Нужны пакеты - gst-python, gst-plugins-ugly, gst-plugins-good-soup, pysetuptools со slackbuilds.org и gst-plugins-base, gst-plugins-good со slackfind.net.

./build в папке с сорцами, далее стандартное создание пакета через make install DESTDIR и makepkg.

Примечание - самый хороший звук по результатам "слепого" прослушивания в версии 1587 из svn, так что если нужен самый качественный звук, то выполняем в терминале

```
svn co -r 1587 \
https://guayadeque.svn.sourceforge.net/svnroot/guayadeque guayadeque
```

Правда интернет радио в этой версии не работает, в этом случае можно использовать следующие программки, первая позволяет узнать url потока а также проигрывать через внешний плеер, остальные играют заданные вручную потоки.

**streamtuner2** — менеджер интернет радио и ТВ.
Сорцы - streamtuner2.sourceforge.net

Нужны либы - pyxdg, lxml, pyquery со slackbuilds.org (поскольку это питон, то слакбилды не правим вообще), keybinder-0.2.2 с kaizer.se/wiki/keybinder.

Поскольку автор streamtuner2 не озаботился созданием установочного скрипта, то пишем простейший свой , streamtuner-setup, кладем его в папку с сорцами, делаем исполняемым и запускаем

```
#!/bin/bash
mkdir -p /tmp/streamnuner2 /tmp/streamnuner2/usr/bin
/tmp/streamnuner2/usr/share /tmp/streamnuner2/usr/share/pixmap
/tmp/streamnuner2/usr/share/streamtuner2
cp ./st2.py /tmp/streamnuner2/usr/bin/
cp ./streamtuner2.png /tmp/streamnuner2/usr/share/pixmap/
cp -r ./* /tmp/streamnuner2/usr/share/streamtuner2/
rm -rf /tmp/streamnuner2/usr/share/streamtuner2/st2.py
rm -rf /tmp/streamnuner2/usr/share/streamtuner2/streamtuner2.png
cd /tmp/streamnuner2/
/sbin/makepkg  -l y -c n /tmp/streamtuner2.txz
rm -rf /tmp/streamnuner2
```

Пакет создается в /tmp.

**radiotray** - проигрыватель интернет радио.
Сорцы — radiotray.sourceforge.net. Также есть слакбилд со slackbuilds.org.

Необходим Cython со slackbuilds.org - (поскольку это питон, то слакбилд не правим вообще).

Впрочем гораздо удобнее консольная обертка к mplayer для проигрывания онлайн-радио — **pyradio** с github.com/coderholic/pyradio, которая просто распаковывается из архива в любое подходящее место. Удобнее тем, что позволяет использовать аудиоэквалайзер mplayer для улучшения далеко не блестящего качества звука онлайн-радио. Для использования аудиоэквалайзера прописываем в ~/.mplayer/config.

```
# Аудио эквалайзер
af=equalizer=6:4:2:0:0:0:0:3:5:8
```

естественно что параметры эквалайзера подбираются по собственному вкусу. Урлы самих станций берем из **streamtuner2**.

Для тех, кто много слушает онлайн может быть интересен проигрыватель **foobnix** — www.foobnix.com, заточенный как раз под онлайн прослушивание, причем в предустановках есть весь guzei.com, множество русских радиостанций, немного со sky.fm и xiph.org, интеграция с ВКонтакте с поиском и скачивания с него и довольно хорошим эквалайзером. Нужна либа simplejson со slackbuilds.org - (поскольку это питон, то слакбилд не правим вообще).
Создание пакета стандартное для питона. Есть слакбилд на www.wuala.com/SergMarkov19/Slackbuilds

**peyote** - консольный проигрыватель.
Сорцы — peyote.sourceforge.net.
Конфигурация по шаблону.
Нужны либы mutagen, pyinotify со slackbuilds.org.
В слакбилде pyinotify комментируем, чтобы было вот так

```
# cp -a ACKS ChangeLog_old COPYING NEWS_old $PKG/usr/doc/$PRGNAM-$VERSION
```

Немного о такой кому необходимой кому нет веши как регулятор громкости в трее. Есть комплект утилит для настройки alsa - qastools со slackbuilds.org, в котором есть такой регулятор, есть отдельные регуляторы, такие как volumeicon, volti и retrovol со slackbuilds.org или fbmix — sandbox.ltmnet.com/fbmix. Последний наверное лучший по простоте, но тут уж на вкус и цвет.

Естественно что аудиоплееров масса, и audacious, и deadbeaf и qmmp и даже xmms, их настолько много что выбор определяется только предпочтениями, которые у каждого свои, поэтому выше и были даны свои предпочтения :-) Но никто не мешает поставить какой угодно плеер, благо большинство из них есть на slackbuilds.org

**feff** — видеоконвертер.
Бинарник — dansoft.krasnokamensk.ru/more.html?id=1013
Сорцы
http://qt-apps.org/content/show.php/Feff?content=140298&PHPSESSID=5e0f8817dce26a9eb7650e499e1c42d7

У кого выше крыши всяких DVD тому нужен каталогизатор дисков, который сканирует их содержимое, хранит и дает возможность искать по содержимому. В принципе таких каталогизаторов вагон и маленькая тележка, но одни тянут полный комплект кедо-гномолиб, другие используют свой формат хранения данных и когда прога прекращает поддерживаться использовать ее данные уже проблематично, еще одни крайне примитивны. Есть прога cdcat без таких недостатков — cdcat.sourceforge.net, она есть на slackbuilds.org, но там старая версия для qt3 и ставить лишний тулкит, к тому же здоровенный, только для одной проги как то не комильфо, поэтому лучше собрать ее самому из последней версии с ее сайта. Для нее нужны либы libzen,libtar, libmediainfo со slackbuilds.org, либа lib7zip со code.google.com/p/lib7zip. Для компиляции lib7zip нужно распаковать сорцы p7zip со slackbuilds.org в папку с lib7zip и запустить в ней make. Далее, поскольку автор lib7zip не соизволил в Makefile сделать секцию install, пакет с ней придется делать ручками простейшим скриптом, запускаемым из папки с сорцами lib7zip

```
mkdir -p /tmp/ lib7zip/usr/include
mkdir -p /tmp/ lib7zip/usr/lib
cp ./Lib7Zip/*.h /tmp/ lib7zip/usr/include
cp ./Lib7Zip/*.o /tmp/ lib7zip/usr/lib
cd /tmp/ lib7zip
/sbin/makepkg -l y -c n /tmp/ lib7zip.txz
```

Компилируем сам cdcat, в cdcat.pro аккуратненько заменяем везде /usr/local/ на /usr/, запускаем qmake cdcat.pro, далее процедура создания пакета, но не через makepkg а через slacktrack. Как именно смотрим как всегда выше. Впрочем на slacky.eu есть готовый пакет, пусть и не самой новой версии.

Конвертировать музыку можно простейшими скриптами, пример которых есть во вложение worker.tbz, а лучшего менеджера тегов чем foobar под wine лучше и не искать, как и лучшей разрезалки cue+ape(flac) чем CUESplitter опять же под wine тоже трудно найти :-)

## 5 - Офис

**libreOffice** - офисный пакет.
Берем здесь — www.libreoffice.org/download.
Архивы *install-rpm-en-US*, *helppack-rpm_ru,langpack-rpm_ru

- Распаковываем все архивы.
- Из папки с распакованным архивом *install-rpm-en-US* временно переносим куда нибудь все файлы *slackbuilds.org*dict*
- Выполняем в этой папке rpm -Uhv *.rpm
- Распаковываем при помощи file-roller архивы *dict*

Или можно просто преобразовать rpm в tgz утилитой rpm2tgz. Также готовые скомпилированные из сорцов пакеты для новых версий libre достаточно быстро выкладываются на alien.slackbook.org/blog. Проверку орфографии смотрим здесь — help.libreoffice.org/Writer/Checking_Spelling_and_Grammar/ru.

Но, как показывает практика, для 95% задач вполне подходит **officekids** :-), который намного легче и быстрее что самого оригинального office что его форка libre. Весь этот текст пишется в нем. Брать officekids здесь - download.ooo4kids.org/ru. Также есть и officelight для студиозов, но в нем нет посконного интерфейса. Устанавливаются они также как и libreoffice.

Проверка русского языка в officekids устанавливается просто - достаточно открыть через "Файл-Открыть" русский словарь extensions.services.openoffice.org/en/project/dictru.

Впрочем после того как openoffice взял под свое крылышко апач, ситуация с ним резко улучшилась. Претензии к нему ранее были в основном в части неимоверной тормознутости. Непонятно что сделали в фонде апача, но теперь openoffice работает быстрее libre, он быстрее запускается, быстрее и скорость открытия доков и скорость интерфейса, экспорт в pdf вообще быстрее в разы. Так что имеет полный смысл вернуться снова к openoffice. Скачиваем нужную версию отсюда — download.i-rs.ru/pub/openoffice, распаковываем архивы tar.gz, преобразуем rpm в tgz утилитой rpm2tgz (rpm2tgz *.rpm), устанавливаем tgz.

Есть еще более простой путь, gnumeric уже давно лучше calc во всех форках OO и в нем самом, abiword более чем подходящ для написания нескольких страничек, а такие тексты составляют подавляющий объем от общего числа, и просмотра остальных. Нет совершенно никакой надобности для написания пары тройки страниц или составления и просмотра простейших табличек запускать этих тормозных монстров. Gnumeric и abiword есть на slackbuilds.org.

Ну а кому не нравится всякие офисы всегда есть lyx :-) со slackbuilds.org, для него нужен tetex с DVD или со slackfind.net. Чтобы lyx подхватил программы из tetex включаем в ~/.bashrc

```
export PATH=$PATH:/usr/share/texmf/bin
```

Немного о такой необходимой вещи как записные книжки, которые постоянно висят в трее и в которые походу кидаются умные (ну или всякие :-) мысли. Их довольно много и выбор опять же определяется требованиями. Кодеру нужна подсветка синтаксиса, тому кто много тянет из инета нужна возможность скопировать страницу из браузеру в записную книжку в первозданном виде, со всеми картинками и прочими плюшками, у кого информации выше крыши уделит внимание не только простой иерархической структуре, но и наличию тегов и особому представлению информации. Далее всякие вики типа zim не рассматриваются, они конечно хороши, но к ним надо привыкнуть. Поэтому на выбор.

**keepnote**

Особенности - позволяет сохранить всю страницу из инета простым копированием из браузера. Поскольку написана на питоне то запускается довольно медленно.
Со slackbuilds.org

**mytetra**

Особенности - в наличии теги и удобное, (в отличии от других) трехпанельное представление.
Сорцы — webhamster.ru/site/page/index/articles/projectcode/105
Заменяем в mytetra.pro

```
BINARY_INSTALL_PATH=/usr/local/bin
```

на

```
BINARY_INSTALL_PATH=/usr/bin
```

Есть слакбилд на www.wuala.com/SergMarkov19/Slackbuilds и slackbuilds.org.

**notecase -** простейшая записная книжка с форматированием и удобным поиском.

Очень быстрая и удобная если не нужны навороты (а в большинстве случаев они вовсе не нужны, это только представляется что без ну никак нельзя, но если посмотреть что из этих наворотов реально используется, то обнаруживается что чуть меньше чем нуль целых хрен десятых :-). Также она позволяет создать исполняемый файл, включающий в себя как саму программу так и ее данные (базу записей). Правда у нее есть глюк - надо выходить из нее перед перезагрузкой иксов или компа, иначе придется восстанавливать последнюю информацию из резервной копии. Сорцы и патчи берем с packages.ubuntu.com и делаем по аналогии с hddtemp, описанной выше.
Правим Makefile, заменяем

```
if [[ "$(TEST_SRCVIEW_VER)" < "2.4.1" ]]; then \
```

на

```
if [[ "$(TEST_SRCVIEW_VER)" < "2.11.0" ]]; then \
```

Что на что заменять становиться понятно при ошибках компиляции с оригинальным Makefile.

Есть масса других записных книжек, модных и не очень, тут уж на вкус и цвет. Впрочем стоит обратить внимание на одну из них, TreeSheets - treesheets.com, с весьма своеобразной идеологией, лежащей в ее основе.

Кому то может понадобится органайзер, их тоже полно, от консольных до самых навороченных, здесь можно обратить внимание на qorganizer — qorganizer.sourceforge.net, достаточно простой но одновременно функциональный органайзер.

Кто любит сохранение информации и планирование в виде карт может быть интересен vym или xmind со slackbuilds.org.

Есть много просмотрщиков pdf в состав слаки уже входит evince, есть и множество других, заслуживает внимания **qpdfview** - launchpad.net/qpdfview. Просмотрщик имеет табы, настраиваемый тулбар и множество других вкусностей. Конфигурация стандартная для qmake, некоторые параметры можно изменить в qpdfview.pri.

,

**kchmviewer -** просмотрщик chm, со slackbuilds.org.

**xchm** - еще один просмотрщик chm, он хуже чем kchmviewer но иногда бывает полезен. Со slackbuilds.org

Также со slackbuilds.org ставим калькулятор **speedcrunch** и словарь **stardict.** Сами словари на sourceforge.net/projects/xdxf/files, но там настолько неудобная классификация, к тому же на первый взгляд не так уж много словарей, то взять расклассифицированные словари можно здесь — dl.dropbox.com/u/75441862/stardict-%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D0%B8.tar.

Впрочем кому нужны словари от lingvo, более подробные но и в чем то менее неудобные именно из-за своей излишней подробности, то можно поставить goldendict, просто преобразовав при помощи packageconverter пакет для бубунты с pkgs.org. Сами словари для lingvo понятно где брать .. :-)

**jpdftweak -** утилита для работы с pdf

Позволяет делать оглавления, вкладывать файлы, делать различное оформление и множество других операций с pdf.
Сорцы - jpdftweak.sourceforge.net
Поскольку java просто распаковать куда нибудь в opt.
Запуск стандартный java -Xmx256M -jar файл.jar

**Adobe Acrobat Reader -** иногда бывает нужен для открытия каких то особых pdf (как эта например :-).

Скачиваем с сайта adobe.com установщик, делаем исполняемым и запускаем от рута в терминале.

## 6 - Игры

Здесь маленькие игрушки, монстров нет

**barrage -** танчики.
Сорцы - lgames.sourceforge.net/index.php?project=Barrage
**ltris** - старый добрый тетрис, со slackbuilds.org.
**lbreakout2** - арканоид, со slackbuilds.org

Аддоны к нему - sourceforge.net/projects/lgames/files/add-ons/lbreakout2

**gtkpool** - простенький биллиард.
Сорцы - archive.ubuntu.com/ubuntu/pool/universe/g/gtkpool

**foobillardplus -** навороченный биллиард.
Сорцы - sourceforge.net/projects/foobillardplus

Можно как преобразовать rpm пакет с помощью утилиты rpm2tgz так и скомпилировать из сорцов. Последнее лучше, так как можно включить оптимизацию для карт Nvidia

```
./configure \
--build=i686-pc-linux-gnu \
--host=i686-pc-linux-gnu \
--prefix=/usr \
--libdir=/usr/lib \
--localstatedir=/var \
--sysconfdir=/etc \
--mandir=/usr/man \
--enable-network=no \
--enable-nvidia=yes \
--enable-sound=yes \
CFLAGS="-O3 -march=native -mtune=native -fomit-frame \
-pointer -pipe -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1" \
CXXFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow \
-falign-jumps=1 -falign-labels=1" \
LDFLAGS="-Wl,-O1 -Wl,--as-needed""
```

**Преферанс**

Под линух нет вменяемого, так что остается только запуск под wine из набора - bridge-preferance.ru/PrefPrograms.htm, лучший наверное до сих пор marriage выпуска 2007 года, или он же, но старый, с www.marriage.ru/winmar/index.htm.

**gambler** - online игры (преф, бридж, масса других). Как убрать ограничение придумать легко :-). Программа (java) - www.gambler.ru/download

**openarena** — шутер.
openarena и еще масса -
packages.zenwalk.org/?b=/extra/games&zversion=current

Для любителей старых досовских игр есть dosbox со slackbuilds.org и фронтенд к нему с массой предустановок для популярных тогда игр, DOSBox Game Launcher — members.quicknet.nl/blankendaalr/dbgl, к которому необходим SDL_sound со slackbuilds.org.

## 7 - Использование программ KDE и GNOME

Легко заметить что все приведенные выше программы не зависят ни от kdelibs ни от gnomelibs, что обеспечивает «реактивность» системы. Если возникнет желание использовать какие то программы, требующие к примеру kdelibs (такие как например choqok, хороший twitter клиент) , то в самом общем случае надо установить следующие пакеты — kdebase, kdelibs, kdebase-runtime, oxygen-icons, strigi, phonon, kde-l10n-ru, polkit-kde, polkit-qt. Если нет желания ручками настраивать конфиги кед (занятие крайне малоувлекательное :-), то придется ставить и kdebase-workspace, где находится systemsettings, после настройки kdebase-workspace можно снести. Сказанное в части набора пакетов справедливо для слаки 13.37, возможно что в будущих версиях потребуется устанавливать другие пакеты, но общий принцип понятен.

# VII - Настройка системы

Заводим нового юзера - adduser, выходим из иксов, logout, логинимся под новым юзером и входим в иксы уже под новым юзером.

## 1 - Настройка шрифтов

Важно — перед тем как что то делать с настройками шрифтов, архивируйте /var/cache/fontconfig,/etc/fonts,~/.fonts,~/.fontconfig,~/.fonts.conf,~/.config/font-manager,~/.config/Trolltech.conf,~/.config/fontgroups.xml, чтобы потом не плакать «пропало все». Архивировать их надо всегда, чтобы когда наковыряли настройки шрифтов так, что они превратились в нечто совсем неудобоваримое или использовали для настройки внешнего вида прог KDE systemsettings, который успешно херачит ваши настройки шрифтов, даже если вы их в systemsettings вообще не трогали.

Настраиваются через font-manager, настройки зависят только от собственных предпочтений и представлений, поэтому далее только личные предпочтения - включить сглаживание и полное (full) уточнение и выключить все остальное.

Иногда рекомендуется вместо редактирования ~/.fonts.conf (посредством font-manager или ручного редактирования) прописать параметры шрифтов в ~/.Xresources, аналогичные значения в нем выглядят так

```
Xft.antialias: 1
Xft.hinting: 1
Xft.hintstyle: hintfull
Xft.lcdfilter: 0
Xft.rgba: 0
Xft.autohint: 0
```

но поскольку восприятия шрифтов, опять же, вещь сугубо индивидуальная, то на взгляд автора шрифты так выглядят хуже. Такое восприятие вполне может быть и плацебо, но в конечном счете при выборе метода отображения шрифтов ориентируемся именно на свое восприятие :-)

Самые азы оптимизации шрифтов для слаки на забугорном изложены здесь - duganchen.ca/writings/slackware/fonts (естественно не надо буквально применять все советы подряд, а смотреть что подходит под свои вкусы). В принципе об их настройке на нормальном :-) уже было сказано выше при описании компиляции freetype. Но там было сказано только об одной части проблемы, различных отрисовках выбранного шрифта при помощи разных патчей и настроек. Но не меньшее значение имеет и выбор самого шрифта, если он изначально коряв, то блестящим он от этих методов не станет. Есть множество всяких шрифтов, и новые от убунту, гугля и paratype, и старые, типа liberation и пакета pscyr. Здесь уж на вкус и цвет. Шрифты из пакета pscyr в формате Type1, поэтому копируем в ~/.fonts все шрифты из пакета, и pfb и afm.

Множество хороших кириллических шрифтов можно найти по этим линкам

free.type.org.ua
www.thessalonica.org.ru/ru/fonts-download.html

connie.slackware.com/~alien/slackbuilds/msofficefonts
pier.botik.ru/~znamensk/ftp.vsu.ru/font-packs/pscyr
www.webtag.ru/fonts
www.dafont.com
briefmobile.com/download-roboto-font-from-android-4-0

Большинство из кучи шрифтов, которые по умолчанию ставятся в слаке, совершенно не нужны и только замедляют систему, в них поддержка и тайского и тамильского и прочей экзотики, которой большинство вряд ли пользуется поэтому удаляем соответствующие пакеты из /var/log/packages. Для гарантии можно сделать mkfontdir и mkfontscale, fc-cache -frv во всех директориях /usr/share/fonts. Чем меньше шрифтов тем быстрее иксы и тем быстрее грузятся многие проги, поэтому можно или выключить или вообще удалить неиспользуемые шрифты. Нельзя удалять только fixed, cursor, cursor.pcf, Liberation, Microsoft Sans Serif, Arial, Tmes New Roman, Terminus и какой нибудь дополнительный serif для офисов, например AdonisC, шрифт для читалки handbookpscyr, DejaVu Sans Mono для терминалов, остальные можно или выключить в font-manager или вообще удалить ненужные пакеты шрифтов, кроме adobe, которые могут потребовать некоторые программы, ну и растровые шрифты естественно нельзя удалять.

При добавлении своих шрифтов в ~/.fonts выполняем в этой папке mkfontdir, mkfontscale, fc-cache -frv и далее настраиваем их через font-manager. Если что то не так со шрифтами, восстановите из архива предварительно сохраненные в нем настройки и перезагрузите комп, это помогает :) . Не сохранили настройки, ССЗБ:-)

## 2 - Сервисы

Находятся в /etc/rc.d. Чем меньше ненужных сервисов тем комп легче дышит :-) В большинстве практических случаях для десктопа для остановки сервиса делаем в терминале сервис stop, и чтобы он не включался при последующих перезагрузках снимаем с соответствующего файла атрибут "исполняемый" хоть в терминале, хоть в mc, хоть в worker. Для включения сервиса делаем его файл исполняемым и командуем сервис start. Для перезапуска, который необходим если что то изменили в конфигурации сервиса (файлы *.conf в /etc), делаем в терминале сервис restart. В более общем случае ищем название сервиса поиском по содержимому файлов в /etc/rc.d и комментируем его вызов.

Совершенно минимальный рабочий набор сервисов, без которых одиночный десктоп просто не будет нормально работать - rc.4, rc.5, rc.K, rc.M, rc.S (которые собственно не сами сервисы а перечень их наборов при разных уровнях запуска компа), rc.alsa, rc.dnsmasq, rc.font, rc.fuse, rc.hald, rc.inet1, rc.keymap, rc.loop, rc.messagebus, rc.modules-*, rc.sysvinit, rc.udev. На отлаженном компе можно выключить и rc.syslog.

Если хочется выполнить что то свое при выходе, создаем /etc/rc.d/rc.local_shutdown, в который пишутся свои команды, которые надо выполнить при выходе, синтаксис такой же как и у /etc/rc.d/rc.local, который выполняет аналогичную функцию при входе. Естественно делаем его исполняемым. Все операции с сервисами делаются от рута.

## 3 - Модули ядра

Убираем оставшиеся лишние модули, комментируя строки с ненужными в /etc/rc.d/rc.modules*. Запускаем hardinfo, в Kernel Modules смотрим что лишнее и вносим их в /etc/modprobe.d/blacklist.conf таким образом

```
blacklist <модуль>
```

В любом случае IPV6 лишнее по любому, так как нехило тормозит инет, и если не убрали его поддержку при конфигурировании ядра, то добавляем в /etc/modprobe.d/blacklist.conf

```
blacklist ipv6
```

## 4 - prelink

Применяется для предварительного связывания программ и библиотек, убыстряет запуск программ

**Не ставьте его на слаку 13.37 - система вообще перестанет загружаться.** Если все же поставили, и система таки не загрузилась, то загружаемся с какого нибудь liveCD, желательно новых версий, распихиваем файлы из пакета prelink в файловую систему LiveCD, правим в ней /etc/prelink.conf чтобы для каждой опции -I было примерно вот так (естественно заменяем на свою точку монтирования слаки)

```
-l /mnt/sda1/bin
```

и запускаем prelink-undo из пакета prelink. Перезагружаемся в слаку (хоть система загрузится) и в ней снова выполняем уже ее prelink-undo. Ну и конечно сразу удаляем пакет prelink :-) Снова перезагружаемся, вуаля, слака как новенькая. Это в лучшем случае, в худшем придется переустанавливать часть пакетов, смотрим при запуске системы на ругань, смотрим на аналогичную ругань при запуске своих программ и переустанавливаем. Вообще то prelink в слаке после 13.1 лучше даже палкой не тыкать, по крайней мере его версии от 2009-2011 годов.

## 5 - sudoers

Вписываем в /etc/sudoers программы, которые хотим запускать от рута при помощи sudo но без ввода пароля рута. Таким образом можно запускать программы от рута например из меню WM, или в каких нибудь скриптах.

Что то типа этого, как пример

```
ed ALL=NOPASSWD: /usr/sbin/hddtemp,/usr/sbin/smartctl
ed ALL=NOPASSWD: /usr/bin/worker,/usr/bin/htop
```

где «ed» - свой юзер.

## 6 - Настройка bash

Настроить его можно под собственные вкусы как угодно, поэтому ниже пример только базовых настроек в ~/.bashrc, которые понадобятся в большинстве случаев.

```
# .bashrc
# User specific aliases and functions
# Source global definitions
if [ -r /etc/bashrc ]; then
 . /etc/bashrc
fi

# Приглашение
#PS1="[\[\033[0m\]\w]\[\033[0m\]\n\[\033[0m\]\u\[\033[0m\]-> \
#[\033[0m\]"
PS1='\[\033[1;34m\]\w\n\[\033[0;30m\]\u->'

#Bash — довольно развитый командный интерпретатор, поддерживающий #кучу
разных настроек Список всех возможных опций можно посмотреть #командой
«shopt -p» (shopt — сокращение от Shell Options).
#Устанавливаются опции следующим образом:
shopt -s autocd cdspell checkjobs cmdhist dirspell globstar

#В историю не будут попадать дубликаты и команды ls, bg, fg,exit
export HISTIGNORE="&:ls:[bf]g:exit"

# Настройка gcc
export CHOST="i686-pc-linux-gnu"
export CFLAGS="-O3 -march=native -mtune=native \
-fomit-frame-pointer -pipe -mmmx -m3dnow -falign-jumps=1 \
-falign-labels=1"
# Иногда приходится использовать стандартные опции gcc,
# закоментированные ниже ниже
#export CFLAGS="-O2 -march=i486 -mtune=i686"
export CXXFLAGS="${CFLAGS}"
export LDFLAGS="-Wl,-O1 -Wl,--as-needed"

# алиасы
alias gdeman="apropos"
alias gde="slocate"


# Убираем использование pango для отрисовки страниц в firefox для
# увеличение скорости отображения страниц
export MOZ_DISABLE_PANGO=1

# Используем raster для рендеринга в qt
export QT_GRAPHICSSYSTEM="raster"


# Программы по умолчанию
export XEDITOR=juffed
```

```
export EDITOR=ne
export VISUAL=$EDITOR
export BROWSER=firefox


# Параметры терминала
export TERM=linux

# Внешний вид java приложений
export _JAVA_OPTIONS='-Dawt.useSystemAAFontSettings=on \
-Dswing.defaultlaf=com.sun.java.swing.plaf.gtk.GTKLookAndFeel'
export JAVA_FONTS=/usr/share/fonts/TTF

# Пути, необходимые для подхвата lyx программ из tetex,
# естественно можно дописать кучу своих путей
export PATH=$PATH:/usr/share/texmf/bin

# Если хочется использовать bash-completion
# Use bash-completion, if available
#if [ -f /etc/bash_completion ]; then
#  . /etc/bash_completion
#fi
```

## 7 - Настройка файловой системы и винтов

- Ускоряем файловую систему.
  Редактируем файл /etc/fstab

```
/dev/sda* / ext3  defaults,noatime,nodiratime,barrier=0,commit=120,data=ordered 1 1
```

где
- noatime,nodiratime - отключение записи времени последнего доступа к файлам и папкам
- barrier - асинхронная запись изменений на диск
- commit - период синхронизации данных и метаданных
- data=ordered - способ синхронизации данных. Не ставьте кое-где рекомендуемый writeback, это отложенная из кэша запись изменений и при случайном сбое в питании изменяемые в этот момент файлы и сохраняемые в кэше могут испортиться. Впрочем если есть хороший UPS (или даже нет его, но очень хочется поиметь себе приключений на определенное место :-) то можно поставить и writeback, но только изменением опций монтирования в fstab он не ставится. Надо загрузиться с какого нибудь LiveCD, отмонтировать свой раздел, на котором будет меняться эта опция, и выполнить команду

```
tune2fs -O has_journal -o journal_data_writeback /dev/sda*
```

после этого исправить в fstab опцию на data=writeback и сплюнув энное количество раз перезагрузиться :-)

Не надо включать индексирование директорий, если у вас нет таковых с

миллионами файлов, оно только замедляет работу на стандартных директориях.

- Ускоряем винты, включая DMA, 32-битный доступ и unmask_irq

```
hdparm -d1c3u1  /dev/винт
```

- Включаем SMART для винтов. В /etc/rc.d/rc.M раскомментировать, чтобы было вот так

```
# Start smartd, which monitors the status of S.M.A.R.T.
# compatible
# hard drives and reports any problems. Note some devices
# (which aren"t
# smart, I guess ;) will hang if probed by smartd, so it"s
#commented
# out
# by default.
if [ -x /usr/sbin/smartd ]; then
        /usr/sbin/smartd
fi
```

Если какое то устройство будет тормозить, как и предупреждают, то снова закомментировать.

- Убираем резервирование 5 % места для рута на не корневых (не / или не /var или не /tmp) разделах linux с ext на отмонтированном (обязательно!!) разделе

```
tune2fs -m 0 /dev/sda*
```

На корневом разделе десктопа 5% для рута тоже совсем не нужно, если у вас на нем не вертится какой то супернагруженный сервер, поэтому можно, загрузившись с какого нибудь LiveCD и отмонтировав раздел со своим корнем, проделать тоже и с корнем, оставив для рута 1 %

```
tune2fs -m 1 /dev/sda*
```

## 8 - Настройки свапа и кэша

- Делаем слаку более отзывчивой, заставив держать как можно больше в памяти, а не сбрасывать в свап. Создаем файл /etc/sysctl.conf и пишем в него

```
vm.swappiness=20
```

- Оптимизируем отношение ядра к освободившимся от кэшированных объектов файловой системы страницам ОЗУ , пишем в /etc/sysctl.conf

```
vm.vfs_cache_pressure = 1000
```

## 9 - Настройки Xorg и задание типа курсора

Создаем в /home/user и в /root директорию .compose-cache, в которой будут кэшироваться программы. Они будут быстрее стартовать и потреблять меньше памяти, благодаря тому, что libX11 будет создавать в ~/.compose-cache кэши распарсенной информации и использовать ее повторно.

Изредка приходится пользоваться такими иксовыми приложениями как xpaint для снятия скриншотов или xfontsel чтобы узнать параметры шрифта, но видок у них такой, что навевает мысли о винде 3.1. В основном это касается совершенно угребищного темно-серого фона и некоторых других вещей. Изменить их можно в файле ~/.Xresources. В нем же задаем тему курсора и его размер, сами темы курсоров кладутся в ~/.icons. Ниже пример такого файла

```
Xcursor.theme: Jimmac
Xcursor.size: 20

##
## xfontsel
##
xfontsel*background:     #FFFBEF
xfontsel*foreground:     #000000
xfontsel*font:   -adobe-helvetica-*-r-*-*-17-*-*-*-*-*-*-*

##
## xpaint
##
xpaint*background:       #FFFBEF
xpaint*foreground:       #000000
xpaint.toolbox.vport.box*background: #FFFBEF
xpaint.toolbox*font:   -adobe-helvetica-*-r-*-*-17-*-*-*-*-*-*-*
Canvas*background:       #FFFBEF
Canvas*foreground:       #000000
Canvas.pane*background:       #FFFBEF
Canvas.pane*foreground:       #000000
Canvas.pane*font:   -adobe-helvetica-*-r-*-*-17-*-*-*-*-*-*-*
```

Как видно из примера после имени окна идет или просто общий для всего окна параметр со значением или имя ресурса окна и уже затем параметр для конкретного ресурса со значением. Имя ресурса для иксовой проги можно узнать при помощи editres.

## 10 — Задание иконок

Для приложений на базе GTK. Распаковываем архив с темой в ~/.icons. Для замены иконок во всех темах пишем в ~/.gtkrc-2.0

```
//название_темы - название каталога с иконками в ~/.icons(не полный   //путь!)
gtk-icon-theme-name="название_темы"
gtk-icon-sizes="gtk-menu=24,24:   //размер значков в меню
```

```
gtk-button=24,24:          //размер значков на "кнопках"
gtk-small-toolbar=32,32:  //размер значков на панельках
gtk-large-toolbar=24,24:
gtk-dnd=32,32:
gtk-dialog=24,24"   //размер значков в диалоговых окнах
```

Для замены иконок в конкретной теме пишем в файл gtkrc в ~/.themes/тема или в /usr/share/themes/тема тоже самое что и для случая всех тем.
Для приложений на базе KDE пишем в файл ~/.kde/share/config/kdeglobals

```
[Icons]
Theme=gnome
```

где gnome имя темы (папки с иконками) в /usr/share/icons

## 11 - Настройки GTK

- Ускоряем GTK приложения, уменьшая время задержки интерфейса.
Добавляем в ~/.gtkrc-2.0

```
gtk-menu-popup-delay = 50
gtk-menu-popdown-delay = 50
gtk-menu-bar-popup-delay = 50
gtk-enable-animations = 0
gtk-timeout-expand = 10
```

- Определяем иконки для GTK приложений
Добавляем в ~/.gtkrc-2.0

```
gtk-theme-name = "Human"
```

где иконки Human находятся в папке ~/.icons/Human или /usr/share/icons/Human
Остальные параметры, которые можно регулировать в ~/.gtkrc-2.0, можно посмотреть в www.gtk.org/api/2.6/gtk/GtkSettings.html

- Настраиваем оставшееся, тему GTK и шрифты для GTK приложений, через gtk-chtheme

## 12 - Настройки QT

- Ускоряем QT приложения, используя для рендеринга raster.
Добавляем в ~/.bashrc

```
export QT_GRAPHICSSYSTEM="raster"
```

– Настраиваем внешний вид и некоторые другие параметры QT приложений через qtconfig. При желании можно вручную покопаться в секции [QT] файла ~/.config/Trolltech.conf, но особого смысла в этом нет, все эти значения проще настроить именно через qtconfig

Естественно для задания разных параметров и внешнего вида GTK и QT при запуске от обычного юзера и рута надо править соответствующие файлы в /home и в /root, и запускать gtk-chtheme и qtconfig от обычного юзера и от рута

## 13 - Включаем NumLock

Есть два способа включения NumLock.

Первый при помощи утилитки numlockx со slackbuilds.org, которая включается вверху скрипта запуска WM /etc/X11/xinit/xinitrc.ваш-WM, следующим образом, после

```
#!/bin/sh
# $XConsortium: xinitrc.cpp,v 1.4 91/08/22 11:41:34 rws Exp $
```

добавляем

```
# Turn NumLock on (using numlockx)
if [ -x /usr/bin/numlockx ]; then
  /usr/bin/numlockx
fi
```

или просто добавляем в скрипт автозапуска WM, например для IceWM это /home/user/.icewm/startup, следующим образом

```
/usr/bin/numlockx &
```

Второй способ начисто вырубает на цифровой клавиатуре все, кроме самих цифр. Создаем файл ~/.Xmodmap и пишем в него

```
keycode 79=7
keycode 80=8
keycode 81=9
keycode 83=4
keycode 84=5
keycode 85=6
keycode 87=1
keycode 88=2
keycode 89=3
keycode 90=0
keycode 91=period
keycode 108=Return
keycode 86=plus
keycode 82=minus
```

```
keycode 63=asterisk
keycode 112=slash
```

## 14 - Графический вход в WM

Если лень набирать startx, изменяем на графический вход, редактируем файл /etc/inittab

```
# These are the default runlevels in Slackware:
# 0 = halt
# 1 = single user mode
# 2 = unused (but configured the same as runlevel 3)
# 3 = multiuser mode (default Slackware runlevel)
# 4 = X11 with KDM/GDM/XDM (session managers)
# 5 = unused (but configured the same as runlevel 3)
# 6 = reboot
# Default runlevel. (Do not set to 0 or 6)
id:4:initdefault:
```

## 15 - Русские имена файлов в zip архивах

Иногда попадаются zip архивы с крякозябрами в именах файлов. Преимущественно это случается с архивами винды в кодировке cp1251, но могут быть и другие кодировки.
Для начала узнаем кодировку

```
unzip -l archive.zip |enca
```

Затем преобразуем в каталоге с распакованным архивом, подбирая кодировки если надо

```
ls | iconv -c -f cp1252 -t cp850 | iconv -c -f cp866 -t utf8
```

Можно также использовать маленькую утилитку convmv - www.j3e.de/linux/convmv, которую кладем в /usr/local/bin. Сначала узнаем кодировку архива, подбирая кодировки если надо

```
$ ls -1 | head -1 | iconv -f CP1251 -t utf8
```

Затем преобразуем в каталоге с распакованным архивом

```
ls -1 | head -1 | xargs convmv -f CP1251 -t utf8 -r --notest
```

Как настроить патчами корректное отображение кириллических имен файлов в архивах zip описано здесь — www.opennet.ru/tips/info/2494.shtml.

Но самый простой способ он как всегда самый лучший :-), благо таких архивов очень мало , просто ставим под wine любой виндовый архиватор – 7z, peazip,

winrar.

## 16 - Другое

- Настраиваем dnsmasq как кэширующий DNS сервер
Вот здесь - http://linuxru.org/man/spravka-po-nastroyke-dnsmasq-dnsmasqconf, dnsmasq.conf с русскими примечаниями.

- В файле /etc/dnsmasq.conf дописываем (или раскомментируем и дописываем) строчку

```
listen-address=127.0.0.1
```

и добавляем в /etc/resolv.conf самой первой строчкой

```
nameserver 127.0.0.1
```

– В настройках браузеров убираем что то типа "предупреждающее чтение dns", поскольку от двойного кеширования только вред. В firefox для этого надо установить в «0» значения network.dnsCacheExpiration и network.dnsCacheEntries в about:config..

- Делаем/etc/rc.d/rc.dnsmasq исполняемым и выполняем

```
/etc/rc.d/rc.dnsmasq start (или restart)
```

- Для блокировки всякой хитрой рекламы и другого нежелательного контента добавляем в секцию

```
# Add domains which you want to force to an IP address here.
# The example below send any host in double-click.net to a local
# web-server.
```

что то типа

```
address=/adfox.ru/127.0.0.1
```

при этом на 127.0.0.1 отправляются не только запросы к самому adfox.ru, но и запросы ко всем его поддоменам и страничкам.

- Если при установке не настроили сеть то сейчас самое время ее настроить

```
netconfig, pppoe, etc
```

- Настраиваем звук если надо

```
alsaconf, alsamixer (от юзера) alsactl store (от рута)
```

- Настраиваем частоту повтора и задержку нажатия клавиш клавиатуры для убыстрения работы с текстом

```
xset r rate 200 40
```

где
200 - задержка в мс
40 - частота повтора Гц

Восстановить значения по умолчанию можно командой xset r rate без параметров

- .. много много тюнинга, который не стоит делать. Чтобы сделать лучше надо знать как параметры оптимизации влияют друг на друга и понимать что именно хочется, иначе вряд ли будет лучше , хуже может стать запросто :-).

# VIII - Настройка оконных менеджеров (WM)

## 1 - Настройка IceWM

### - Настройка основных параметров

Айс легкий, красивый и широко настраиваемый, но его внешний вид и настройки по умолчанию могут отпугнуть сразу, почему автор годами не хочет их менять это уже вопрос к нему.

Настраивается ручной правкой конфигов в ~/.icewm. Поскольку после установки их там нет, то копируем в ~/.icewm умолчальные настройки из /usr/share/icewm, файлы keys, menu, preferences, toolar и winoptions.

В начале читаем описания настроек IceWM, первое старое, но базовые принципы не изменились

docstore.mik.ua/manuals/ru/icewm/icewm-ru.html#toc6
quickcode.chat.ru/icewm/icewm-ru.html
mydebianblog.blogspot.com/2006/10/icewm.html.
avreg.net/howto_icewm.htm
konishchevdmitry.blogspot.com/2008/07/icewm.html
vectorlinux.osuosl.org/docs/vl50/vlfaq/icewm.htmt.

Естественно что самая подробная документация на самом icewm.org, но даже там автор, прекрасный программер, но бооольшой лентяй :-) ее не обновляет уже бог знает сколько времени.

Основные настройки редактируются ручками в файле preferences. Настройки с подробными комментариями, так что разобраться в них не составляет никакого труда. Настроить его можно как угодно, читаем комменты в preferences (если хватит терпения :-) или ищем поиском в нем (что гораздо лучше) что именно надо настроить.

Вот это настраиваем сразу, так как сильнее всего пугает при первом взгляде на IceWM

```
#  Menus track mouse even with no mouse buttons held (меню без
# щелчка)
MenuMouseTracking=1 # 0/1

#  Support win95 keyboard keys (Penguin/Meta/Win_L,R shows menu)
Win95Keys=1 # 0/1

#  Support mouse wheel
UseMouseWheel=1 # 0/1

#  Alt+Tab window switching
QuickSwitch=1 # 0/1

#  Alt+Tab to windows on other workspaces
QuickSwitchToAllWorkspaces=1 # 0/1

#  Opaque window move
OpaqueMove=0 # 0/1
```

```
#  Opaque window resize
OpaqueResize=0 # 0/1
```

- Настраиваем меню в menu, для примера (начинающиеся с runonce и prog в одной строке)

```
menu "Графика" "folder" {
    runonce "Geeqie - Просмотрщик изображений" "/usr/share/pixmaps/geeqie.png"
"geeqie" geeqie
separator
    prog "Screengrab - Скриншот"
 "/usr/share/pixmaps/FBReader/rtf.png" screengrab
}
```

где runonce не позволяет запускать вторую копию уже запущенной программы.

- Настраиваем тулбар с выпадающим меню в toolbar (начинающиеся с prog в одной строке)

```
prog "xterm" "xterm" xterm
prog "juffed - Редактор" "/usr/share/pixmaps/juffed.png" juffed
prog "Worker - Файловый менеджер" "/usr/share/pixmaps/WorkerIcon16.xpm" worker
prog "firefox" "/usr/lib/firefox-4.0b6/chrome/icons/default/default16.png" firefox
menu "Система" "folder" {
   prog "Htop-root - Менеджер процессов" "/usr/share/pixmaps/htop.png" roxterm --tab
--tab-name=htop -e sudo htop
   prog "Aumix - Регулятор громкости" "/usr/share/aumix/aumix.xpm" aumix
}
```

– Настраиваем автозагрузку при старте IceWM в startup, как пример

```
#!/bin/sh
compton &
sleep 2
gxneur &
sleep 3
kill `ps -A|awk '/xneur/{print $1}'`
gxneur &
sleep 3
claws-mail &
devilspie &
sleep 2
mytetra &
stardict &
sleep 3
worker &
sleep 2
xdotool search --class Worker windowactivate --sync
xdotool key alt+F9
roxterm &
sleep 3
xdotool search --class RoxTerm windowactivate --sync
```

```
xdotool key alt+F9
cairo-dock -o &
exit
```

где sleep необходимые задержки перед выполнением команд , xdotool сворачивает окна worker и roxterm на панель задач, двойной запуск gxneur необходим из-за его глюка, иначе он не переключает языки для набранного слова. Делаем startup исполняемым.

- Настраиваем расположение и, по желанию, другие параметры окон некоторых выбранных приложений в winoptions. Например

```
# geometry
evince.geometry: 1100x1020+250+10
```

Название и класс окон узнаем при помощи xprop.

- Настраиваем шорткеи в keys. Для примера

```
key "Ctrl+Space"                    /usr/bin/gmrun
```

Для наладки и просто посмотреть что получилось удобно запускать нужный WM в том же сеансе что и уже запущенный WM через Xnest - xorg в отдельном окне

```
Xnest :20 & xterm -display :20
```

откуда из xterm запускать нужный WM. Для повторного запуска надо удалить файлы /tmp/.X20-lock и /tmp/.X11-unix/X20.

**- Темы IceWM**

Тем айса вагон и маленькая тележка, сейчас они находятся в основном на box-look.org (немного тем а-ля WinXP есть на lxp.sourceforge.net), но не сказать что их там очень много, поэтому архив с некоторыми темами выложен на dropbox и wuala, включать его как вложение накладно, архив слишком объемный.

Сами темы находятся в ~/.icewm/themes/папка с темами, для своих скачанных тем разумеется. Настройка темы осуществляется как правкой файла default.theme так и правкой соответствующих рисунков. Сами рисунки, составляющие схему, не описаны в default.theme, их назначение жестко задается названием рисунка. Подробно описано здесь - www.icewm.org/themes. Для первоначальной настройки темы смотрим на эти параметры

```
DesktopBackgroundImage="background.jpg" - картинка фона
```

и задание шрифтов

```
# Font Specification
TitleFontNameXft        =       sans-serif:size=14:bold
MenuFontNameXft                 =       sans-serif:size=14:bold
```

```
MinimizedWindowFontNameXft = sans-serif:size=14
ActiveButtonFontNameXft = sans-serif:size=14:bold
NormalButtonFontNameXft = sans-serif:size=14
ToolButtonFontNameXft = sans-serif:size=14
NormalWorkspaceFontNameXft = sans-serif:size=14
ActiveWorkspaceFontNameXft = sans-serif:size=14:bold
QuickSwitchFontNameXft = sans-serif:size=14
ListBoxFontNameXft = sans-serif:size=14
StatusFontNameXft = sans-serif:size=14
ToolTipFontNameXft = sans-serif:size=14
ActiveTaskBarFontNameXft = sans-serif:size=14:bold
NormalTaskBarFontNameXft = sans-serif:size=14
ClockFontNameXft = sans-serif:size=14
InputFontNameXft = sans-serif:size=14
```

названия остальных параметров интуитивно понятны. Пример настройки IceWM во вложении icewm.tbz.

Немного об оптимизации тем IceWM. Темы для него есть самые разные, от простейших текстовых до навороченных с переходами, переливами, закруглениями и прочими красивыми плюшками. Все это очень хорошо, но на самых навороченных схемах все эти красивости могут тормозить. В основном тормоза связаны с использованием в темах метода gradient, который позволяет изменять размеры и обеспечивает плавный переход от одного рисунка к другому. В default.theme в этом случае присутствует строчка типа такой

```
Gradients="menubg.xpm taskbuttonactive.xpm menusel.xpm dialogbg.xpm
workspacebuttonactive.xpm taskbuttonbg.xpm taskbuttonminimized.xpm"
```

Что в этом случае можно сделать, убрать (закомментировать) саму эту строчку в default.theme и подогнать размеры рисунков темы под друга друга и под ваш монитор . Причем ширину рисунков лучше делать кратные 4, т.е шириной 4,8 пикселей, вместо размеров в 1 пиксель, это убыстряет отрисовку. Надо чтобы размеры элементов темы, описанные в default.theme типа высоты заголовка окна TitleBarHeight=26 соответствовали размеру рисунка. Некоторая заминка может быть связана с шириной рисунков меню, в этом случае меряем с точностью до пикселя ширину корневого меню и делаем рисунки меню точно такой же, до пикселя, ширины. Увеличивает скорость отрисовки элементов темы оптимальный выбор метода отрисовки, самый быстрые это pixmap и flat, определяется в default.theme строчкой типа Look= pixmap.

Еще больше оптимизировать тему можно удалением некоторых ее элементов, если не используем какие то свои рисунки для часиков в трее, то можно удалить папку ledclock, если вполне достаточно стандартных иконок папок и файлов по умолчанию то не нужна папка icons, если не используете нотификацию почты в трее (допустим в трее и так уже висит почтовик со своей нотификацией, к тому же нотификацию на gmail в айсе настроить нельзя) то удаляем папку mailbox, если вполне хватает системных шрифтов и курсоров то не нужны папки fonts (естественно надо в этом случае переопределить шрифты на системные в default.theme) и cursor. Также можно удалить файлы dframe* и frame*, отвечающие за отрисовку окантовки окна и заменить их простым заданием цвета окантовки и ее ширины в default.theme. В папке taskbar за глаза хватает следующих рисунков - desktop.xpm icewm.xpm, taskbarbg.xpm taskbuttonactive.xpm,

taskbuttonbg.xpm, taskbuttonminimized.xpm.

Вообще говоря темы зачастую перегружены какими то старыми версиями рисунков и прочими ненужными файлами, которые автор поленился удалить, вычищаем и их.

## 2 - Настройка openbox

Копируем умолчальные настройки, файлы в /etc/xdg/openbox в /home/user/.config/openbox. Файл rc.xml служит для настройки внешнего вида окон, их поведения и способов управления ими. Кроме того, в файле определяются необходимые пользователю сочетания клавиш. Файл menu.xml определяет содержимое различных меню. Любые меню (в том числе и вложенные) могут иметь идентификаторы, что позволяет вызывать их различными сочетаниями клавиш. Скрипт autostart исполняется во время загрузки оконного менеджера. Таким образом можно инициализировать переменные окружения и запускать дополнительные приложения. Настраивается или ручной правкой конфигов или через гуи, сам openbox через obconf, меню через obmenu, темы через obtheme и/или menumaker.

Lxpanel настраивается конфигами в /home/user/.config/lxpanel/default. Умолчальные конфиги в /usr/share/lxpanel/profile/default. Tint2 настраивается через гуй tint2conf. Для работы tint2conf делаем симлинк из /sbin/killall5 в /usr/bin/pidof.

Почти исчерпывающая инструкция по настройке openbox вот здесь wiki.debian.org/ru/Openbox или здесь wiki.archlinux.org/index.php/Openbox или на забугорном вот здесь – urukrama.wordpress.com.

## 3 - Настройка Enlightenment

Тут писать вообще то нечего, настройка осуществляется через простейший гуй, как настроить меню и некоторые другие вещи можно прочитать здесь sda00.blogspot.com и здесь radist-elvin.blogspot.com/search/label/enlightenment

Есть подробный цикл публикаций о E17 на rus-linux.net/nlib.php?name=/MyLDP/gui/E17/e17-index.html

Вообще есть rus-linux.net/main.php?name=x-win.ko#6.4., где можно найти многое о менеджерах окон и их настройке.

Поскольку стандартные настройки переключения клавиатур в E17 не действуют (по крайней мере в его версии на момент написания), то вносим в автозагрузку E17.

```
setxkbmap -option "grp:ctrl_shift_toggle,grp_led:scroll"
```

Темы для E16 — themes.effx.us/previews
Темы для E17 — e17-stuff.org

## 4 - Настройка композита

Естественно заменить один оконный менеджер другим просто нельзя, но можно использовать некоторые эффекты (тени окон, эффекты меню) со своим оконным менеджером при помощи xcompmg или compton (отличие compton от xcompmgr смотрим в его доках). Как именно хорошо описано в wiki.archlinux.org/index.php/Xcompmgr. Обратите внимание на настройку xorg.conf. Xcompmgr requires the following: Xorg must be installed, configured and running. Composite must be enabled via graphics drivers, AIGLX, or Xgl. Xcompmgr процессор практически не нагружает, если, конечно, не повключать все эффекты подряд и радостно наблюдать за тормозами . Вообще то использовать Xcompmgr для каких то других эффектов кроме теней под окнами лучше не стоит, на других эффектах в зависимости от компа он может тормозить как буратино.

Для использования и оптимизации xcompmg добавляем в секцию секцию Section "Device" xorg.conf

```
Option    "RenderAccel" "true"
Option    "AllowGLXWithComposite" "true"
```

и добавляем в конец xorg.conf

```
Section "Extensions"
    Option "Composite" "Enable"
    Option "RENDER"    "Enable"
EndSection
```

С какими параметрами запускать и какие эффекты как использовать смотрим как всегда xcompmgr —help и курим man xcompmgr, аналогично для compton. Но лучше не использовать все эффекты подряд а ограничиться простейшими типа теней под окнами - xcompmgr -c, иначе может тормозить.

Для включения композита при загрузке IceWM в /etc/X11/xinit/xinitrc.icewm заменяем

```
exec /usr/bin/icewm-session
```

на

```
/usr/bin/xcompmgr -c &
exec /usr/bin/icewm-session
```

аналогично в других оконных менеджерах.

Быстрое включение и отключение композита можно сделать скриптом. Скрипт кладется в /usr/local/bin и прописывается в меню и/или забиндивается на какую то клавишу.

```
#!/bin/bash
#
# Start a composition manager.
# (xcompmgr in this case)
function comphelp () {
  echo "Composition Manager:"
```

```
  echo "   (re)start: COMP"
  echo "   stop:      COMP -s"
  echo "   query:     COMP -q"
  echo " returns 1 if composition manager is running, else 0"
  exit
}
function checkcomp () {
  ( ps nc -C xcompmgr &>/dev/null ) && exit 1
  exit 0
}
function stopcomp () {
  ( ps nc -C xcompmgr &>/dev/null ) && killall xcompmgr
}
function startcomp () {
  stopcomp
# Example settings only. Replace with your own.
  hsetroot -solid "#FFFBF0" &
  xcompmgr -cC &
  exit
}
case "$1x" in
    "x") startcomp;;
  "-qx") checkcomp;;
  "-sx") stopcomp; exit;;
     *) comphelp;;
esac
```

# IX - Настройки отдельных программ

### worker

Вход в гуй настроек по кнопочке "C" в левом верхнем углу. Настройка worker в связке с roxterm и juffed, позволяющая использовать worker одновременно и в root и в user mode, позволяющая также легко собирать пакеты задача долгая и нудная, поэтому во вложении worker.tbz приведен пример такой настройки. Естественно меняем разрешение файлов на свои. Может потребоваться изменить что то несущественное в файлах, типа комментариев, и уже затем изменить разрешение.

О выборе шрифтов в worker, по сути у него единственный выбор из ttf шрифтов это sans-serif, значение которого определяется в файле /etc/fonts/conf.avail/60-latin.conf, секция sans-serif (по порядку описания в нем). Как его можно изменить смотрим чуть выше в разделе 'Улучшение шрифтов' .

Worker везде очень активно использует правую кнопку, и в панели файлов и в кнопках и везде где только можно, так что если чего то в нем не хватает, жмем правую кнопку и скорее всего нужное появится :-)

### gxneur (xneur)

У него есть такие возможности как самообучение и автодополнение, которые лучше выключить в силу несовершенства алгоритма их реализации. Если их включить, то можно наткнуться на такую «особенность», что все двух-трехбуквенные слова или предлоги будут автоматом переводить на забугорный, то есть напечатанное по русски «но» будет автоматом переводиться в «yj». Если все таки включили эти возможности и напоролись на такие «особенности», то надо очистить содержимое файлов ~/.xneur/pattern и ~/.xneur/dictionary и выключить все таки эти возможности в настройках gxneur.

### cairo-dock

Если появляется черный фон панели вместо прозрачного, включаем в настройках "Система" - "Эмулировать композитность с поддельной прозрачностью". В icewm для такой эмуляции обязательно должна быть установлена картинка на рабочий стол, иначе появляется черный фон в cairo-dock.

При запущенном композитном менеджере типа xcompmgr или compton можно запускать cairo-dock с поддержкой openGL - cairo-dock -o. Работает быстрее, но и больше жрет памяти.

Чтобы не запускать из cairo-dock множество копий одной и той же программы, если она уже запущена, в общем случае делаем небольшой скрипт

```
#!/bin/bash
#
# Скрипт проверки запущена ли программа и если "да" то сделать ее окно
# активным, если "нет" то запустить
#
APP=`basename $1`
FULLAPP="$*"
HOSTNAME=`hostname`

# пробуем найти уже запущенный экземпляр программы
```

```
# и вывести ее окно на передний план
wmctrl -l -x | grep -i $APP | while read RUNNING
do
        WINDOW=${RUNNING#*${HOSTNAME} }
        wmctrl -a $WINDOW
        exit 1
done

# запускаем программу, если она еще не запущена
if [ $? -eq 0 ]
then
    $FULLAPP
fi
```

для worker и juffed скрипт будет выглядеть несколько по иному

```
#!/bin/bash
#
# Скрипт проверки запущена ли программа и если "да" то сделать ее окно
# активным, если "нет" то запустить
# модификация для worker и juffed
#
APP=`basename $1`
FULLAPP="$*"
HOSTNAME=`hostname`

# пробуем найти уже запущенный экземпляр программы
# и вывести ее окно на передний план
wmctrl -l -x | grep -i $APP | while read RUNNING
do
        WINDOW=${RUNNING#*${HOSTNAME} }
        wmctrl -a $1
        exit 1
done

# запускаем программу, если она еще не запущена
if [ $? -eq 0 ]
then
    $FULLAPP
fi
```

В оба скрипта в качестве параметра передается имя программы, а сам вызов скрипта осуществляется в настройках соответствующего значка запуска cairo-dock.

Как пример скрипт запуска ночнушки firefox

```
#!/bin/bash
#
# Скрипт проверки запущена ли программа и если "да" то сделать ее окно
# активным, если "нет" то запустить
#
```

```
APP=`basename "Nightly"`
FULLAPP="firefox"
HOSTNAME=`hostname`

# пробуем найти уже запущенный экземпляр программы
# и вывести ее окно на передний план
wmctrl -l -x | grep -i $APP | while read RUNNING
do
        WINDOW=${RUNNING#*${HOSTNAME} }
        wmctrl -a $WINDOW
        exit 1
done

# запускаем программу, если она еще не запущена
if [ $? -eq 0 ]
then
   $FULLAPP
fi
```

Пример настроек во вложении cairo-dock.tbz. Естественно меняем разрешение файлов на свои.

**gsmartcontrol**

В настройках надо выставить путь к smartctl - /usr/sbin/smartctl

**luckybackup**

Для сохранения предыдущих копий файлов ставим в параметрах задачи значение "Snapshot to keep" больше нуля, сколько хотим сохранить предыдущих копий. Сохраненные предыдущие копии находятся в папке .luckybackup-snaphots.

**psensor**

Определяем установленные сенсоры, запускаем от рута

```
/usr/sbin/sensors-detect
```

**wine**

Запускаем в терминале winecfg и настраиваем.

Сглаживание шрифтов в wine (скрипт) - www.wine-reviews.net/wine-reviews/tips-n-tricks/how-to-enable-font-anti-aliasing-in-wine.html

Если есть винды делаем симлинк из WINDOWS/Fonts в /home/user/.wine/drive_c/windows.

Далее добавляем при помощи winetricks необходимые либы винды.

Для запуска программ с параметрами, например для вызова из worker с именем файла, делаем скриптик наподобие этого

```
env WINEPREFIX="/home/ed/.wine" wine \
"C:\Program Files\Tracker Software\PDF-XChange Viewer\pdf-viewer\PDFXCview.exe" \
"$(winepath -w "$@")"
```

Небольшая но емкая хаутушка по wine — zenway.ru/page/wine-howto

**xonclock**

Настройки в ~/.xonclockrc. Вырубаются двойным щелчком правой кнопкой мышки.

**google-chrome**

Добавляем в самый верх /opt/google/chrome/google-chrome
cd /opt/google/chrome
и делаем симлинк с него в /usr/local/bin.

К хрому полно плагинов, которые выбираем на вкус и цвет :-), но есть прекрасный плагин, позволяющий сохранить страничку со всем содержимым (картинками, скриптами итп) в одном html-файле - SingleFile и SingleFile Core.

Некоторые плагины хрома при обновлении выключаются и если обновления не очень нужны (как правило это так и есть :-) и лень каждый раз включать плагины, то убираем автообновление - делаем неисполняемым файл /etc/cron.daily/google-chrome.

Для сохранения зрения от изысков уэбдизайнеров и убыстрения работы создаем файл /home/user/.config/google-chrome/Default/User StyleSheets/Custom.css и пишем в него

```
* { text-align: justify !important; }
* { line-height: 1.40 !important; }
* { font-family: Liberation Sans !important; }
* {
border-radius: 0 !important;
box-shadow: none !important;
}
```

Естественно это только пример своего css, тут уж на вкус и цвет :-)

Для более точной настройки размеров шрифтов, прописываем в /home/user/.config/google-chrome/Default/Preferences что типа этого , размеры естественно ставим свои

```
 "webkit": {
   "webprefs": {
      "default_fixed_font_size": 21,
      "default_font_size": 24,
      "fixed_font_family": "Liberation Sans",
      "minimum_font_size": 21,
      "serif_font_family": "Liberation Sans"
   }
```

**firefox**

По аналогии с хромом для убережения своего зрения ликвидируем изыски уэбдизайнеров, пишем в файл (естественно это только пример своего css) /home/ed/.mozilla/firefox/*.default/chrome/userContent.css

```
* { text-align: justify !important; }
* { line-height: 1.25 !important; }
* { font-family: Liberation Sans !important; }
* {
border-radius: 0 !important;
box-shadow: none !important;
}
```

Также более точно определяем размер минимального шрифта, поскольку в гуи настроек фокса слишком большой шаг между возможными значениями такого размера, для этого правим в файле /home/user/.mozilla/firefox/*.default/prefs.js значения следующих параметров, опять же ниже только пример значений, они подбираются по вкусу

```
user_pref("font.minimum-size.x-cyrillic", 21);
user_pref("font.minimum-size.x-western", 21);
```

Плагинов к фоксу немеряно, у каждого свой набор, но для любителей ЖЖ есть плагин deepestsender-deepestsender.mozdev.org.

Также могут понадобится flashvideoreplacer, проигрывание флеша с youtube.com через xine, и downloadhelper для скачивания видео. Проигрывание браузерного флеша через mplayer, при использовании которого можно регулировать видеопараметры, возможно посредством следующего скрипа

```
#! /bin/bash
# Проигрывание флеша через mplayer
for i in `ps ax | grep flash | grep -v grep | awk '{print $1}'`; do
        for j in `ls -la /proc/$i/fd | grep /tmp/Flash | awk '{print $9}'`; do
                mplayer6 /proc/$i/fd/$j
        done
done
```

флеш на страничке запускается, ставится на паузу и запускается этот скрипт.

Естественно приведены расширения помимо стандартных adblock, umht и кучи всего другого :-)

Для использования самых новых версий фокса понадобится расширение Nihtly Tester Tools, которое приспосабливает расширения от старых версий к версии новой.

Для использования firefox в связке с кэширующим DNS сервером dnsmasq устанавливаем в 0 значения network.dnsCacheExpiration и network.dnsCacheEntries в about:config.. Для корректного копирования кириллицы из адресной строки firefox (которая есть например в вики) устанавливаем в false значение network.standard-url.escape-utf8.

Для подключения дополнительный протоколов типа magnet временно устанавливаем в about:config параметр network.protocol-handler.expose-all в false, жамкаем какую то magnet ссылку, выбираем приложение, ставим галку в диалоге выбора «по умолчанию», возвращаем значение параметра network.protocol-handler.expose-all в true.

**dillo**

Для включению куков создаем файл /home/user/.dillo/cookierc и пишем в него DEFAULT ACCEPT.

Поскольку dillo в основном используется для просмотра локальных файлов нет смысла в ухищрениях уэбдизайнеров и лучше смотреть сохраненные странички в нормальном виде (впрочем это касается и браузеров для просмотра инета :-), поэтому создаем файл /home/user/.dillo/style.css и пишем в него

```
body {background-color: white !important}
body {color: black !important}
:link {color: blue !important}
* {
font-family : LiberationSans ! important; flashvideoreplacer
font-size : 20px ! important;
}
```

**licq**

Для корректного подсоединения может понадобится установить порт подключения через login.icq.com.

**webhttrack**

Для вызова firefox вместо умолчального chrome заменяем в /usr/bin/webhttrack (SRCHBROWSEREXE в одной строке)

```
SRCHBROWSEREXE="x-www-browser 5www-browser iceape mozilla firefox icecat iceweasel abrowser firebird galeon konqueror opera netscape"
```

на

```
SRCHBROWSEREXE="firefox icecat iceweasel abrowser firebird galeon konqueror opera netscape"
```

и заменяем

```
SRCHBROWSEREXE="xdg-open sensible-browser ${SRCHBROWSEREXE}"
```

на

```
SRCHBROWSEREXE="firefox"
```

**officekids**

Мелкие шрифты интерфейса лечатся установкой нужного масштаба шрифта в /home/user/.ooo4kids/1.0/user/registry/data/org/openoffice/Office/Common.xcu

```
<prop oor:name="FontScaling" oor:type="xs:short">
 <value>160</value>
</prop>
```

**libreoffice, officekids, juffed - плагины**

Ненужные плагины замедляют запуск и саму работу с прогами, поэтому убираем лишние плагины, для libreoffice создаем папку /opt/libreoffice3.4/share/extensions-old копируем в нее содержимое /opt/libreoffice3.4/share/extensions, затем в /opt/libreoffice3.4/share/extensions удаляем папки с лишними плагинами. По аналогии удаляем ненужные плагины в officekids и juffed (плагины в usr/share/juffed/plugins).

Разумеется кроме ненужных есть и нужные плагины :-), для офисов есть к примеру плагин **ooo2gd** для работы с google docs (не сказать что он очень удобен, но может быть полезен) - extensions.services.openoffice.org/project/ooo2gd

**clipit**

В нем есть интересная особенность, "Automatically paste selected item", которая позволяет прямую вставку в нужное окно, без "вставить" в этом окне, только выбором нужного пункта в меню clipit, но она некорректно работает со всеми офисами, clipit в этом случае впадает в ступор и помогает только его убиение через htop и перезапуск, так что лучше этой возможностью не пользоваться и убираем поэтому ее в настройках.

**FBReader -** для читалки критичен хороший шрифт, есть масса serif, но наверное лучший это handbookpscyr из пакета pscyr с pier.botik.ru/~znamensk/ftp.vsu.ru/font-packs/pscyr. Этот шрифт разновидность Baltica, но на больших размерах смотрится лучше ее.

**mplayer**

Пример конфига ~/.mplayer/config

```
[default]

# Видеодрайвер
vo=xv,noslices

# Аудиодрайверы ( в порядке перебора)
ao=alsa,oss

# Пропуск кадров
framedrop=true

# Постобработка (%)
autoq=100

# Кэш
cache=2048

# Кодовая таблица субтитров
subcp=cp1251

#Пропускает фильтр loop (он же deblocking) во время декодирования H.264.
lavdopts=skiploopfilter=nonref
```

```
#Перестраивает индекс, если необходимо
idx=1

# Аудио экстрастерео
af=extrastereo=2.00

# Имя файла в заголовке окна
use-filename-title=on

#####################################ed##################
######################

# Аудио эквалайзер
# af=equalizer=6:4:2:0:0:0:0:3:5:8

#Постепенно подстраивает A/V синхронизацию на основе измерений задержки
аудио.
#autosync=30

[gnome-mplayer]
vo=xv
ao=alsa
alang=Russian,rus,ru
msglevel=all=5
slang=Russian,rus,ru
af=equalizer=6:4:2:0:0:0:0:3:5:8
vf=eq2
```

Чтобы не запоминать множество опций проще создать пару тройку скриптов со стандартными опциями размытия (резкости) типа таких

```
k=`echo $1|sed s/' '/'-'/g`
mv -T "$1" $k
# Запуск mplayer с фильтрами резкости для нерезких фильмов
# unsharp=lc3*3:1.5.0.0.0 регулируется значение 1.5, чем больше тем резче
# mplayer -af equalizer=6:4:2:0:0:0:0:3:5:8 -vf eq2,pp=l5,unsharp=lc3*3:1.5.0.0.0,hue $k
mplayer -vf eq2,pp=l5,unsharp=lc3*3:0.5.0.0.0,hue $k
```

и

```
k=`echo $1|sed s/' '/'-'/g`
mv -T "$1" $k
# Запуск mplayer с фильтрами размытия для шумных и резких фильмов (осторожно -
сильное потребление процессора)
# hqdn3d убрать шум
# smartblur=1:0.4:0 размытие (0.4 сила размытия, чем больше тем выше)
# Образец
# mplayer -vf eq2,hue,pp=l5,hqdn3d,smartblur=1:0.7:0 $1
mplayer -vf hqdn3d,eq2,hue,smartblur=1:0.3:0 $k
```

**foobillardplus**

Поскольку русской локали в нем нет, а без нее при запуске он делает ручкой, создаем файл /opt/games/foobillardplus/bin/foobillardplus.sh и пишем в него

```
cd /opt/games/foobillardplus/bin
LANG=C ./foobillardplus
```

естественно делаем его исполняемым и запускаем foobillardplus из него.

Для быстрого запуска программ KDE удобно сделать предзагрузку компонент кед, включив запуск kdeinit4 в автозагрузку WM.

Некоторые особо умные авторы делают в новых версиях полностью или частично несовместимые настройки со старыми версиями, из-за чего прога начинает глючить и работать вовсе не по привычному. Тогда полностью удаляем настройки старой версии и настраиваем прогу по новой

В финале архивируем всю эту радость (раздел со слакой) чем угодно, лучше qt4-fsarchiver, на другой винт или раздел. Впрочем еще лучше архивировать qt4-fsarchiver со сжатием lzo отдельные директории (/bin, / boot, / etc, /home, /lib, /opt, /root, /sbin, /usr, /var , остальные архивировать не нужно) на другой винт или раздел, поскольку можно быстро и просто восстановить только нужное. Нельзя только архивировать при помощи qt4-fsarchiver ntfs разделы и создавать архивы на них, могут быть проблемы.

# XI - Ежедневное обслуживание

Слака_как и любая система требует ежедневного обслуживания, такого как очистка системы от всякого хлама и бэкапы.

## 1 - Очистка системы

Всякий хлам удаляется при помощи bleachbit простейшим скриптиком, который засовывается в меню IceWM

```
#! /bin/bash
bleachbit
sudo bleachbit
```

естественно сам bleachbit должен быть прописан в  /etc/sudoers.

В процесса работы может возникнуть необходимость очистить память, свап и кэши, чтобы не тормозило. Также делается простейшим скриптиком,запускаемым от рута, естественно скриптик прописывается в /etc/sudoers. Чтобы очистился свап, закрываем программы, которые его в этот момент могут использовать.

```
#!/bin/bash
#
# Скрипт очистки корзины, кэша google chrome,
#  кэша изображений, сборок в /tmp и свапа
#

echo "Сколько было свободно на диске до очистки"
df -h /
echo ""
echo "Начинаем очистку"
rm -rf /.trash
mkdir /.trash

# Заменяем пользователя ed на своего
# и путь к кэшу фокса и хрома на свой

chown ed:users /.trash
echo "Корзина очищена"
rm -rf /home/ed/.cache/google-chrome/*
echo "Кэш google-chrome очищен"
rm -rf /home/ed/.mozilla/firefox/k3pcop68.default/Cache/*
echo "Кэш firefox очищен"
rm -rf /tmp/SBo
rm -rf /tmp/compile-am
rm -rf /tmp/temp-packages19
echo "Результаты компиляций очищены"
rm -rf /tmp/.avfs*
rm -rf /tmp/.com.google*
rm -rf /tmp/package*
echo "/tmp очищен"
```

```
echo ""
echo "Сколько стало свободно на диске после очистки"
df -h /
echo ""
echo ""
echo "Сколько было свободной памяти"
free
echo ""

# Чистим pagecache:
echo "Сихронизация и очиска системных кэшей"
sync
echo 1 | sudo tee -a /proc/sys/vm/drop_caches

#Чистим dentrie и inode кэши:
sync
echo 2 | sudo tee -a /proc/sys/vm/drop_caches

#Чистим pagecache, dentrie и inode кэши:
sync
echo 3 | sudo tee -a /proc/sys/vm/drop_caches

echo "Синхронизация и очистка системных кэшей выполнена"
echo ""
echo "Очищаем свап"

# Подправить под свой раздел со свапом

/sbin/swapoff /dev/sda9
/sbin/swapon /dev/sda9
echo "Очистка свапа завершена"
echo ""
echo "Сколько стало свободной памяти"
free -m
echo "Все очистки завершены"
```

Скрипт прописывается в меню IceWM вот так (prog в одной строке)

```
prog "Очистить корзину, /tmp, все кэши и свап" "/usr/share/pixmaps/xchat.png" roxterm
--tab --tab-name=clear-trash-cache-tmp -e sudo скрипт
```

Также весьма полезно периодически чистить ручками /home от конфигов удаленных прог, неиспользуемых тем, курсоров и иконок и прочего мусора, который не удаляет bleachbit. Такая очистка повышает быстродействие системы.

Если понаставляли недавно для пробы и интереса всякого мусора, открываем в файл-менегере папку /var/log/packages, сортируем ее по времени и удаляем этот хлам.

## 2 - Бэкапы

Полезность бэкапов своих файлов и системных настроек звучит везде постоянно как мантра, но что это на самом деле правда, как правило понимаешь только после того как сам получишь граблями по лбу :-) Согласно пословице «Умный учится на своих ошибках, дурак на своих», даже если не принимать во внимание сами ошибки и связанные с ними хлопоты, то лучше выглядеть умным, по крайней мере в своих глазах и хотя бы поэтому :-), делать бэкапы все же крайне рекомендуется.

Сначала о бэкапе системы, одна ее часть это бэкапы настроек, другая полный бэкап всей системы.

Бэкапы настроек в /etc, /home, /root, /var делаем ежедневно вручную или по cron при помощи luckybackup от рута со включенными snapshot (чтобы сохранить разные версии). Бэкапы делаем куда нибудь на другой раздел, а еще лучше на другой винт и/или флешку. Периодически архивируем и скидываем бэкапы на любое файлохранилище в инете (amazon cloud drive или любое другое).

Бэкап системы можно сделать partimage (partclone), но они делают образ системы в одном файле и не получится восстановить из него на меньший раздел чем оригинальный,. Поэтому лучше использовать qt4-fsarchiver, с его возможностью сделать как образ системы в одном файле так и архивировать каждую системную директорию (/bin, / boot, / etc, /home, /lib, /opt, /root, /sbin, /usr, /var , остальные архивировать не нужно) в свой файл с выбранной степенью сжатия. Архивацию и восстановление директорий можно производить даже на действующей системе, которая используется в этот момент. Самый быстрый метод сжатия lzo, сжимает примерно в 2-2.5 раза, но зато работает очень быстро.

Бэкапы пользовательских файлов, тут уж на вкус и цвет, важно лишь пользоваться всеми методами, как делать бэкапы куда нибудь в инет (dropbox, wuala, google docs) при помощи указанных выше соответствующих прог так и делать локальные бэкапы при помощи luckybackup на другой винт (раздел) и/или на флешку, причем делать такие бэкапы каждый день. Ну и естественно включать создание резервных файлов во всех программах,который умеют это делать, место для таких резервных файлов лучше конечно выбирать на другом винте (разделе).

Немного об облаках, как широко распространенном сегодня месте хранения бэкапов. Есть облака имеющие свои linux клиенты, это dropbox.com, wuala.com,spideroak.com, для других, как google docs (drive), есть сторонние клиенты типа SuperFlexibleSynchronizer — superflexible.com, еще для одних типа Яндекс.Диск (help.yandex.ru/disk/?id=1124655#programs) есть доступ по webdav при помощи Konqueror, Nautilus или davfs2 со slackbuilds.org (как именно хорошо описано в документации в сорцах и в unihub.ru/resources/38), есть также облака с хорошим web-интерфейсом типа adrive.com (50 Гб) или Amazon Cloude Drive — amazon.com/clouddrive, тут уж на вкус и цвет.

## 3 - Другое

**Обновление кэша иконок и шрифтов**

Оно делается при каждой загрузке, но некоторые проги (типа zim) весьма вольно обращаются с иконками и шрифтами, в частности zim при деинсталляции корежит кэш иконок. Для таких случаев может быть полезен простейший скриптик, который обновляет кэши иконок и шрифтов (запускать от рута)

```
echo "Старт Создание кэша иконок и шрифтов"
#обновление кэша иконок в своей папке
for d in ~/.icons/*; do gtk-update-icon-cache -f $d; done
#обновление кэша иконок в системе
for d in /usr/share/icons/*; do sudo gtk-update-icon-cache -f $d; done
#обновление кэша шрифтов
fc-cache -fv
#/home/user/.fonts заменяем на свою папку шрифтов в хомяке
fc-cache -fv /home/user/.fonts
echo "Окончание Создания кэша иконок и шрифтов"
```

**Оптимизация SQLite**

Базы SQLite имеют обыкновение распухать от ненужного, поэтому оптимизируем их скриптиком (полезно для ff, кед, гнома и гуглехрома)

```
#!/bin/bash
find ~/ -size +100k -type f -print0 | \
while read -d '' FILE; do
  abs_file_name=$(readlink -f "$FILE")
  headfile=`head -c 15 "$abs_file_name"`;
  if [ "$headfile" = "SQLite format 3" ]; then
    file_size_do=`du -b "$abs_file_name"|cut -f1`;
    sqlite3 "$abs_file_name" "VACUUM; REINDEX;" > /dev/null 2>&1
    file_size_posle=`du -b "$abs_file_name"|cut -f1`;
    echo "$abs_file_name";
    echo "Размер ДО $file_size_do";
    echo "Размер ПОСЛЕ $file_size_posle";
    echo -n "Процент "
    echo "scale=2; ($file_size_posle/$file_size_do)*100"|bc -l
  fi
done
sleep 2
exit 0
```

Выходим из иксов (Ctrl-Alt-Backspace), запускаем скрипт

```
optimizsqlite.sh > ~/report-optimizsqlite.txt
```

перезагружаемся и смотрим чего он там наоптимизировал в ~/report-optimizsqlite.txt.

# Линки

## Сайты Slackware

www.slackware.com
slackworld.berlios.de/links.html – множество линков на документацию и пакеты
www.slackware.ru
www.slackguide.com
optimization.hardlinux.ru – сайт об оптимизации linux

## Готовые пакеты и slackbuilds

slackbuilds.org
www.wuala.com/SergMarkov19/Slackbuilds – слакбилды для некоторых программ, которые есть в этом описании, но для которых нет слакбилдов на slackbuilds.org
slacky.eu
slackfind.net
connie.slackware.com/~alien/slackbuilds
rlworkman.net/pkgs
slackers.it
packages.zenwalk.org/?v=current

## Поиск пакетов, в том числе по аналогии

slak.homelinux.org
www.z01.eu/slak
www.teoxonline.com/utils/sse
slakfinder.frattocchie.it/slak
pkgs.org
packages.ubuntu.com

## Документация

slackbook.org (англ)
www.opennet.ru/docs/RUS/slackware (рус)
jack.kiev.ua/docs/slackbook (рус)

## Патчи к ядру

ck.kolivas.org/patches
ck-hack.blogspot.com
algo.ing.unimo.it/people/paolo/disk_sched/patches
pf.natalenko.name
git.zen-kernel.org/zen-stable

## Программы

Qt-Apps.org
GTK-Apps.org

KDE-Apps.org
KDE-Files.org
GnomeFiles.org
Java-Apps.org
Wine-Apps.org
CLI-Apps.org
Qt-Prop.org
Server-Apps.org
apps.ownCloud.com

### Форумы

linuxforum.ru
unixforum.org
www.slackware.ru
www.linuxquestions.org (англ)

### Оформление

themes.effx.us
E17-Stuff.org
exchange.enlightenment.org
KDE-Look.org
GNOME-Look.org
Xfce-Look.org
Box-Look.org A
Beryl-Themes.org
Compiz-Themes.org
EDE-Look.org
Debian-Art.org
Gentoo-Art.org
SUSE-Art.org
Ubuntu-Art.org
Kubuntu-Art.org
LinuxMint-Art.org
Arch-Stuff.org Art
Frugalware-Art.org
Fedora-Art.org
Mandriva-Art.org
VLC-Addons.org

### Темы Icewm

Неоптимизированные
www.box.net/shared/j87dcxouvy2g5cd1zi81

Оптимизированные
www.box.net/shared/1g1u4eguyemtucdm8n4h
www.box.net/shared/yycg8kr6tlsqmjl3axvh
www.box.net/shared/71r6sxvtp91v44kklxlu

www.box.net/shared/ot0co3pp4bhldpz71tgq
www.box.net/shared/t859tdjnak8omvv5z2jm

Более новые версии оптимизированных для разрешения 1920x1080 тем на www.wuala.com/SergMarkov19/Guide-files. Вполне возможно что они подойдут и для других разрешений.

Для оптимизированных тем удаленные части тем находятся в папке тема/remove

Предложения и замечания на sergmarkov1960@gmail.com (c)