**Лабораторная работа №5**

**Цель:** получить навыки работы с текстовым редактором vi, архиватором tar и пакетами в Linux Slackware

**Задачи:**

1. Изучить теоретический материал

**Теоретическая часть**

**Текстовый редактор vi**

vi(1) это стандартный текстовый редактор Unix, который используется в основном системными администраторами. Есть несколько версий (клонов) vi: vi, elvis, vile, и vim. Один из них есть практически в любой версии Unix, и Linux. Все эти версии включают идентичный основной набор команд, так что изучив основные команды одного из клонов, вам будет легко перейти к использованию другого. В Linux есть более простые редакторы, например, nano, или ee (easy editor) в FreeBSD, или встроенный редактор в MidnightCommander. НО, освоив vi вы сможете на любой \*Nix системе редактировать файлы. Например, урезанные версии linux в маршрутизаторах (BusyBox) имеют лишь vi.

vi включает множество мощных вспомогательных инструментов, включая подсветку синтаксиса, форматирование кода, мощный поиск-замена механизм, макросы и многое другое. Эти инструменты делают программу особенно привлекательной для программистов, вэб разработчиков и т.д. Системные администраторы оценят возможный автоматизм и интегрированность с оболочкой.

В Slackware Linux, версия vi по умолчанию, это elvis. Другие доступные версии, такие как vim и gvim, доступны, если вы установили соответствующие пакеты. gvim это X Window версия vim, с панелями инструментов, всплывающими меню и диалогами.

vi может быть запущен из командной строки множеством различных способов:

$ vi

Такая команда запустит vi с пустым буфером. Вы увидите почти пустой экран. Программа сейчас в ''командном режиме'' и ожидает, пока вы сделаете что -либо. Чтобы выйти из vi напечатайте:

:q

Если вы ничего не изменяли в открытом файле, это приведёт к выходу из программы. Если же вы изменили что либо, программа скажет вам, что содержимое файла было изменено и подскажет вам, что напечатать, чтобы выйти, проигнорировав все изменения.

Вы можете так же открыть при помощи vi уже существующий файл. Например, чтобы открыть /etc/resolv.conf, вам надо выполнить:

$ vi /etc/resolv.conf

Так же, vi может открыть файл с указанной строчки. Например, вы можете запустить vi со строки 47 /usr/src/linux/init/main.c, выполнив такую команду:

vi +47 /usr/src/linux/init/main.c

vi выведет на экран указанный файл, и установит курсор на указанную строку. В случае, если номер строки больше номера последней строки файла, vi поместит курсор на последнюю строку. Это очень удобно для программистов, так как они могут открыть файл со строки, в которой была обнаружена ошибка, и таким образом, отпадает необходимость поиска строки вручную.

**Режимы vi**

vi оперирует в разных режимах, которые используются для выполнения различных задач. Когда вы только запустили vi, вы попадаете в командный режим. Отсюда вы можете выполнять разные команды для манипулирования текстом, перемещаться внутри файла, сохранить, выйти и это ещё не всё. Для изменения текста используется режим вставки (insert). Вы можете переключаться в различные режимы, при помощи комбинаций клавиш, которые описаны ниже.

***Командный режим***

Вначале вы попадаете в командный режим. Из этого режима вы не можете напрямую вводить текст, или редактировать уже существующий текст. Но вы можете манипулировать текстом, искать, выйти, сохранить, загружать другие файлы.

Вероятно, наиболее часто используемая команда командного режима, это переход в режим вставки. Нажмите i и вы попадёте в режим вставки. Курсор изменит свою форму и ''- INSERT -'' высветится в нижней части экрана (в клонах программы это не произойдёт). В этом режиме вы можете вводить текст, и он будет отображаться на экране. Чтобы вернуться в командный режим, нажмите Esc кнопку.

Командный режим так же удобен для перемещения внутри файла. На некоторых системах вы можете использовать стрелки для перемещения. В других системах вам придётся воспользоваться более традиционными ''hjkl''. Вот список того, как эти кнопки используются для перемещений:

h – перейти влево на один символ

j – перейти вниз на один символ

k – перейти вверх на один символ

l – перейти вправо на один символ

Просто нажмите на соответствующую букву, для перемещения необходимом направлении. Как будет показано позже, эти команды могут использованы в комбинации с числом, для более эффективного перемещения.

Многие команды, используемые в командном режиме, начинаются с двоеточия. Например, для выхода используется :q, как уже упоминалось выше. Двоеточие просто показывает, что это команда, в то время, как ''q'' говорит vi что вы хотите покинуть программу. Другой тип команд представлен комбинацией числа (необязательного) и буквы. Перед такими командами не надо указывать двоеточие, они используются в основном для манипулирования текстом.

Для примера, чтобы удалить одну строку из файла, нажмите dd. Что приведёт к удалению строки, на которой находится курсор. Выполнение команды 4dd укажет vi удалить строку, на которой находится курсор, и три строки после неё. В общем, число указывает vi, сколько раз выполнить команду.

Вы можете комбинировать число с командами перемещения. Например, 10k переместит вас на 10 строк вверх по тексту.

Командный режим так же может быть использован для вырезания и вставки текста, вставки текста, для считывания других файлов в текущий буфер. Копирование текста осуществляется при помощи y кнопки (y от yank). Копирование текущей строки выполняется нажатием yy и может быть использовано с предшествующим числом, для копирования нескольких строк. Затем перейдёте к месту, куда хотите вставить скопированный текст, и нажмите p. Текст будет вставлен в строку, следующую за текущей.

Вырезание текста выполняется при помощи dd и p используется для вставки вырезанного текста обратно в файл. Считывание текста с другого файла, это довольно простая процедура. Просто наберите :r, пробел и имя файла, содержащего текст, который вы хотите вставить. Содержимое файла будет помещено в текущий буфер, со строки следующей за той, на которой находится курсор. Более усовершенствованные клоны vi завершают имена файла, на подобие того, как это работает в оболочке.

Последний вариант использования режима, который будет описан здесь, это поиск. Режим команд позволяет выполнять как простой поиск, так и расширенный поиск с заменой. В этом разделе будет описан только команда простого поиска.

Для поиска нажмите на кнопку / и введите текст, который вы хотите найти. vi будет искать от того места, где находится курсор, в направлении к концу файла, и остановится при нахождении первого совпадения. Заметьте, что не точные совпадения так же заставят vi остановиться. Например, поиск ''the'' заставит vi остановиться на ''then'', ''therefore'' и т.д. Это произойдёт потому, что все эти слова содержат ''the'', но только в начале.

После того, как vi нашёл первое совпадение, вы можете продолжить, поиск следующего, простым нажатием / и ввод. Можно так же задавать поиск в направлении от курсора, к началу файла, для этого вместо слэш вам надо воспользоваться ?. Например, для поиска в обратном направлении ''the'' вам необходимо выполнить команду ?the.

***Режим вставки***

Вставка и замена текста осуществляется в режиме вставки. Как сообщалось выше, для перехода в этот режим из командного, вам надо нажать i. Далее, весь напечатанный вами текст вводится в текущий буфер. Нажатие Esc вернёт вас в командный режим.

Замена текста осуществляется несколькими способами. Нажатие r из командного режима позволит вам заменить один символ, подсвеченный курсором. Просто введите новый символ и он займёт место подсвеченного курсором. Сразу же после этого вы вернётесь в командный режим. Нажатие R позволяет вам заменить столько символов, сколько вы пожелаете. Чтобы выйти из режима замещения, просто нажмите Esc и вы попадёте в командный режим.

Есть так же и другой способ переключения между вставкой и заменой. Нажатие кнопки Insert из командного режима переведёт вас в режим вставки. Если вы уже в режиме вставки, кнопка Insert работает, как переключатель между режимом вставки и замены.

**Открытие файлов**

vi позволяет открывать файлы как из командного режима, так и из командной строки, указав имя файла после имени программы. Чтобы открыть файл /etc/lilo.conf:

:e /etc/lilo.conf

Если вы произвели изменения в текущем буфере и не сохранили их, vi скажет вам об этом. Вы всё же можете открыть файл, без записи текущего буфера, набрав :e!, пробел и имя файла, который вы хотите открыть. Обычно предупреждения программы могут быть проигнорированы указанием восклицательного знака после команды.

Если вы хотите перечитать текущий файл, вы можете сделать это, набрав e!. Это удобно, если вы каким-то образом что-то напортили и хотите перечитать файл.

Некоторые vi клоны (например, vim) позволяют открыть одновременно несколько буферов. Например, чтобы открыть файл 09-vi.sgml в моём домашнем каталоге, в то время, как другой файл уже открыт, я наберу:

:split /09-vi.sgml

Новый файл отображается в верхней половине экрана, а старый файл отображается в нижней половине. Есть много команд для управления разделённым экраном, и многие из них напоминают EMACS. Лучшей ссылкой по использованию этих команд будет man страница для вашего vi клона.

**Сохранение файлов**

Для сохранения файлов в vi может быть использовано несколько разных команд. Если вы хотите сохранить текущий буфер в файл randomness, вам надо набрать:

:w randomness

После того, как вы однажды сохранили файл, для повторной записи в него, вы можете просто набрать :w. Все изменения сохранятся в файл. После сохранения файла, вы попадаете в командный режим. Если вы хотите сохранить изменения и выйти, из vi (что случается довольно часто), вам надо набрать :wq. Это укажет vi сохранить изменения и выйти.

Иногда необходимо сохранить файл, помеченный только для записи. Вы можете сделать это, указав восклицательный знак после команды записи, вот так:

:w!

:wq!

Все же может случится, что вы не сможете записать изменения, которые вы произвели в файл (например, если файл принадлежит другому пользователю). Если это случится, vi сообщит вам, что он не может записать файл. Если вы действительно хотите изменить файл, вам придётся вернуться и изменить его, как root.

**Выход из vi**

Один из методов выхода из vi это использование :wq, что приведёт к сохранению текущего буфера перед выходом. Вы так же можете выйти без охранения изменений, при помощи :q или :q!. Последний вариант используется когда вы изменили файл, но не хотите сохранить эти изменения.

Может случиться, что произойдёт сбой при работе вашего компьютера, или программы vi. Тем не менее, оба клона (и elvis и vim) предпримут шаги по минимизированию потерь всех открытых буферов. Оба редактора сохраняют открытые буфера во временный файл. Этот файл обычно называется по аналогии с открытым файлом, но с точкой вначале. Это делает файл скрытым.

Этот автоматически удаляется, как только вы нормально завершили редактирование файла. Это значит, что если произошёл сбой, то резервный файл всё ещё будет существовать. И если вы захотите опять редактировать файл, программа спросит вас, как поступить. В большинстве случаев большая часть вашей не записанной работы может быть восстановлена. elvis так же пошлёт вам письмо (из Graceland, что достаточно странно :) сообщающее о существующей резервной копии.

**Настройка vi**

Выбранный вами vi клон может быть настроен несколькими способами.

Множество команд может быть введено в командном режиме для настройки практически всего, что угодно. В зависимости от вашего редактора, вы можете активизировать функции, упрощающие программирование (такие, как подсветку синтаксиса, авто-отступы, и более), установите макрос для автоматизирования задач, активизации подстановки текста, и более.

Почти все из этих команд могут быть помещены в настроечный файл в вашем домашнем каталоге. elvis ищет .exrc файл, а vim ищет .vimrc файл. Большинство конфигурационных команд, которые могут быть введены из командной строки, могут быть размещены в настроечный файл. Включая установочную информацию, подстановку текста, макросы, и более.

Обсуждение всех этих опций и различий между редакторами, не является целью данной книги, так что если вы заинтересовались расширенными функциями редакторов, загляните в man страницу или зайдите на вэб сайт предпочитаемого вами редактора. Некоторые редакторы (как vim) содержат мощную справочную систему, доступную внутри редактора, по команде :help. Вы так же можете заглянуть в книгу ''Изучаем Редактор vi'' Ламба и Роббинса.

Многие программы в Linux по умолчанию откроют текстовые файлы в vi. Например, редактирование ваших crontab задач загрузит vi по умолчанию. Если вам не нравится vi и хотите, чтобы другая программа запускалась по умолчанию, всё что вам надо сделать, так это установить значение VISUAL переменной окружения равное имени вашего любимого редактора. Для информации по установке переменных окружения смотрите раздел 5.1.3. Если вы хотите, чтобы ваш любимый редактор был редактором по умолчанию при каждом входе в систему, добавьте установку VISUAL переменной в ваш .bash\_profile или .bashrc файл.

**Кнопки vi**

Этот раздел является чем-то вроде быстрого справочника многих основных vi команд. Некоторые из которых уже упоминались выше, а некоторые будут для вас новыми.

|  |  |
| --- | --- |
| **Операция** | **Кнопка** |
| **Перемещения** | |
| влево, вниз, вверх, вправо | h, j, k, l |
| перейти в конец строки | $ |
| перейти в начало строки | ы |
| перейти в конец файла | G |
| перейти в начало файла | :1 |
| перейти к 47-й строке | :47 |
| **Редактирование** | |
| удалить строку | dd |
| удалить пять строк | 5dd |
| заменить символ | r |
| удалить символ | x |
| удалить десять символов | 10x |
| отменить последнее действие | u |
| объединить текущую строку со следующей | J |
| **Поиск** | |
| найти äsdf" в тексте после курсора | /asdf |
| искать äsdf"в тексте до курсора | ?asdf |
| повтор последнего поиска в тексте после курсора | / |
| повтор последнего поиска в тексте до курсора | ? |
| **Сохранение и выход** | |
| выход | :q |
| выход без записи | :q! |
| записать и выйти | :wq |
| записать, без выхода | :w |
| пере-загрузить текущий файл | :e! |
| записать содержимое буфера в файл asdf | :w asdf |
| открыть файл hejaz | :e hejaz |
| загрузить файл asdf в буфер | :r asdf |
| загрузить вывод ls в буфер | :r !ls |

**Tar**

tar (англ. tape archive) — формат битового потока или файла архива, а также название традиционной для Unix программы для работы с такими архивами. Программа tar была стандартизирована в POSIX.1-1998, а также позднее в POSIX.1-2001. Первоначально программа tar использовалась для создания архивов на магнитной ленте, а в настоящее время tar используется для хранения нескольких файлов внутри одного файла, для распространения программного обеспечения, а также по прямому назначению — для создания архива файловой системы. Одним из преимуществ формата tar при создании архивов является то, что в архив записывается информация о структуре каталогов, о владельце и группе отдельных файлов, а также временны́е метки файлов.

Как и другие утилиты Unix, tar — специализированная программа, которая следует философии Unix «делать только одну вещь» (в данном случае — работать с архивами формата tar), «но делать её хорошо». Поэтому tar не создаёт сжатых архивов, а использует для сжатия внешние утилиты, такие, как gzip и bzip2. Ранее для сжатия использовалась также утилита compress, которая практически вышла из употребления.

Для файлов, содержащих архивы tar (хотя, tar не сжимает данные, часто архив tar называют «тарбол», по аналогии snowball - снежок), традиционно применяется расширение имени файла .tar. В случае дополнительного сжатия другими программами применяются следующие расширения:

**gzip:** .tar.gz, .tgz (в случае ограничений файловой системы на длину расширения), .tar.gzip

**bzip2:** .tar.bz2, .tar.bzip2, .tbz2, .tb2, .tbz

**compress:** .tar.Z, .taz

**LZMA:** .tar.lzma

**XZ:** .tar.xz, .txz

**lzop:** .tar.lzo, .tzo

**lzip:** .tar.lz, .tlz

**Синтаксис и примеры**

tar [-опции] <имя файла tar> [файлы, которые необходимо поместить в архив при сжатии]

Основные опции:

-c, --create — создать архив;

-r, --append — добавить файлы в конец существующего архива;

-x, --extract, --get — извлечь файлы из архива;

-f, --file — указать имя архива;

-j, --bzip2 — cжать/распаковать архив дополнительно при помощи bzip2;

-z, --gzip, --gunzip, --ungzip — сжать/распаковать архив дополнительно при помощи gzip;

-v, --verbose — выводить список обработанных файлов.

Создание bar.tar из файла или каталога foo:

tar -cvf bar.tar foo

Извлечение содержимого bar.tar в текущий каталог:

tar -xvf bar.tar

**Tarpipe** — процесс создания tar-архива в stdout и последующего извлечения содержания архива в другом каталоге из stdin. Это очень удобный метод копирования содержания одной файловой системы в другую, так как происходит полное копирование, включая атрибуты файлов, символические ссылки, файлы устройств.

tar -C "каталог\_источника" -cf - . | tar -C "каталог\_назначения" -xvf -

tar предоставляет множество полезных возможностей. Например, можно указать файлы и каталоги, которые не будут включены в архив, добавить файлы в существующий архив, взять список объектов для запаковки из текстового файла и много что ещё. Во всем многообразии опций как всегда поможет разобраться

man tar

или же

tar --help

**Установка ПО**

Рано или поздно пользователь столкнется с необходимостью собирать программу из исходников. Всё довольно просто:

./configure

make

make install

Способ хорош в том случае, если нужно собрать программу «под себя», указав специфические ключи для скрипта configure. **НО**, настройка и компиляция может идти долго, в системе может не оказаться другого ПО необходимого для работы, кроме того, дальнейшее сопровождение различного софта, установленного подобным способом, может превратиться в головную боль. Вот поставили вы таким образом сотню программ, а теперь попробуйте удалить 50 из них. В лучшем случае вам поможет скачка той же версии исходников, что и установленная программа, ./configure с теми же параметрами, make и только затем make uninstall, и то если разработчик позаботился о деинсталляции своего творения. В худшем случае – искать по всем папкам вашей системы и вычищать руками все файлы, связанные с программой.

**Управление пакетами Slackware**

Пакет программ это упакованный набор связанных программ, которые готовы для установки. Когда вы скачиваете архив исходных текстов программ, вам надо конфигурировать, компилировать и установить всё вручную. С пакетом программ это уже было проделано за вас. Всё, что вам надо сделать, это установить пакет. Другое очень удобное преимущество пакетов в том, что их можно очень легко удалить или обновить, если вы пожелаете. Slackware поставляется со всеми необходимыми пакетами для управления пакетами. Вы можете устанавливать, удалять, обновлять, создавать и проверять пакеты очень легко.

**Форматы пакетов**

Перед тем, как приступить к изучению утилит, вам надо ознакомиться с форматом пакетов Slackware. Пакет, это всего навсего tar архив, сжатый при помощи gzip. Пакет устроен таким образом, что он может быть развёрнут в корневом каталоге файловой системы.

Здесь приведена вымышленная программа и пример её пакета:

./

usr/

usr/bin/

usr/bin/makehejaz

usr/doc/

usr/doc/makehejaz-1.0/

usr/doc/makehejaz-1.0/COPYING

usr/doc/makehejaz-1.0/README

usr/man/

usr/man/man1

usr/man/man1/makehejaz.1.gz

install/

install/doinst.sh

Система управления пакетами развернёт этот файл в корневом каталоге, чтобы установить его. Будет сделана соответствующая запись в базе данных пакетов, включающая содержание этого пакета, чтобы он мог быть обновлён или удалён позже.

Обратите внимание на под-каталог install/. Это специальный каталог, который может содержать скрипт, который будет выполнен после копирования файлов. Скрипт называется doinst.sh. Если система обработки пакета найдёт такой скрипт, он будет выполнен после установки пакета.

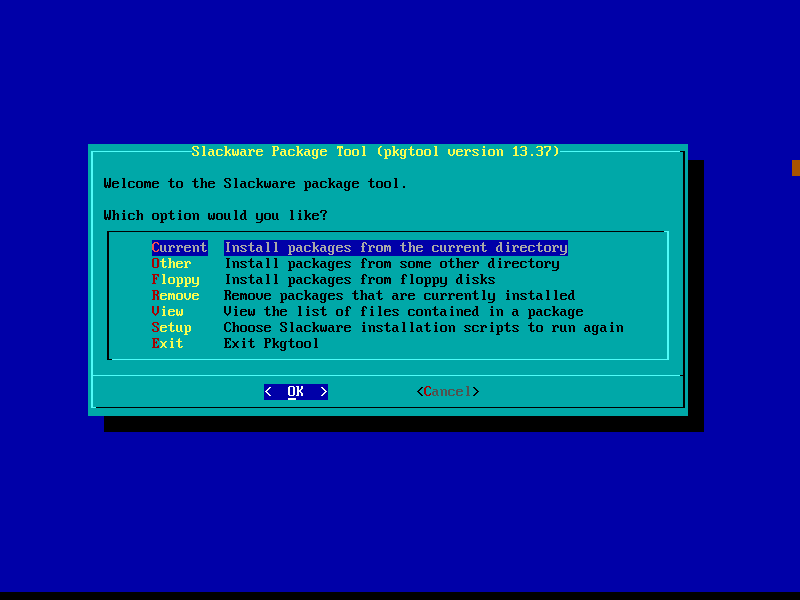
Утилиты пакетов

Есть четыре основных утилиты для управления пакетами. Они могут устанавливать, удалять и обновлять пакеты.

**pkgtool**

pkgtool(8) это управляемая меню программа, позволяющая установку и удаление пакетов. Основное меню выглядит вот так:

Установка может быть произведена из текущего каталога, другого каталога, или с флоппи дисков. Просто выберите метод, который вы хотите использовать и pkgtool просмотрит выбранное вами расположение, на факт наличия действительных пакетов для установки.



Если вы хотите удалить пакеты, выберите remove опцию, и вы увидите список всех установленных пакетов, с возможностью выбора нескольких из них для удаления. Пометьте те, которые вы хотите удалить, и нажмите OK. pkgtool удалит их.

Некоторые пользователи предпочитают эту команду утилитам командной строки. Тем не менее, следует отметить, что утилиты командной строки предоставляют гораздо больше опций. Так же, возможность обновлять пакеты реализована только, как утилита командной строки.

**installpkg**

installpkg(8) обеспечивает установку новых для вашей системы пакетов. Синтаксис таков:

# [ROOT=<path>] installpkg [option] <package name>...

installpkg имеет три опции. Только одна из них может использована при вызове программы.

|  |  |
| --- | --- |
| **Опция** | **Действие** |
| -m | Производит makepkg операцию с текущим каталогом. |
| -warn | Показывает, что случится, если вы установите пакет. Это довольно полезно, так как вы видите в точности, что случится при установке пакета. |
| -r | Установит рекурсивно все пакеты из текущего каталога. <package name> может так же быть маской имён. |

Если вы указали переменную окружения ROOT, до выполнения installpkg, то путь, указанный в переменной будет использован вместо корневого каталога. Это полезно для настройки новых дисков для вашего корневого каталога. Они обычно будут смонтированы к /mnt или что-то другое, отличное от /.

База данных установленных пакетов есть не что иное, как набор файлов, расположенных в /var/log/packages. Для каждого из пакетов там создаётся текстовый файл. Если в пакете есть после-установочный скрипт, он записывается в /var/log/scripts/<packagename>.

Вы можете указать несколько пакетов, или даже использовать шаблоны для имён пакетов. Обратите внимание, что installpkg не скажет вам, если вы переписываете уже установленный пакет. Она просто установит новый пакет поверх старого. Если вы хотите быть уверенны, чтобы файлы от старого пакета были удалены из вашей системы, воспользуйтесь upgradepkg.

**removepkg**

removepkg(8) обеспечивает удаление пакетов, уже установленных в вашей системе. Синтаксис таков:

# [ROOT=<path>] removepkg [option] <package name>...

Для removepkg есть четыре опции. Только одна из них может быть использована при вызове программы.

|  |  |
| --- | --- |
| **Опция** | **Действие** |
| -copy | Пакет будет скопирован в резервный каталог. Это создаст дерево каталогов оригинального пакета, без удаления его. |
| -keep | Сохраняет временные файлы, которые создаются при удалении. Полезно только для тестирования правильности работы. |
| -preserve | Пакет удаляется из системы, но в то же время, его копия сохраняется в резервный каталог. |
| -warn | Показывает, что произойдёт, если вы удалите пакет. |
| -copy | Пакет будет скопирован в резервный каталог. Это создаст дерево |

Если вы указали переменную окружения ROOT, до выполнения removepkg, то путь, указанный в переменной будет использован вместо корневого каталога. Это полезно для настройки новых дисков для вашего корневого каталога. Они обычно будут смонтированы к /mnt или что-то другое, отличное от /.

removepkg просматривает так же и остальные пакеты, и удаляет только файлы, являющиеся уникальными для удаляемого пакета. Программа так же просмотрит пост-установочный скрипт пакета, и удалит все символические ссылки, которые были созданы им.

Во время процесса удаления отображается статус. После удаления, база данных пакета перемещается в /var/log/removed\_packages, а пост-установочный скрипт, в /var/log/removed\_scripts.

Как и в случае с installpkg, вы можете указывать несколько пакетов, или использовать шаблоны для имён пакетов.

**upgradepkg**

upgradepkg(8) обновит пакет, уже присутствующий в вашей системе.Синтаксис программы таков:

# [ROOT=<path>] upgradepkg <package name>...

или

# [ROOT=<path>] upgradepkg \ <old package name>%<new package name>

upgradepkg вначале устанавливает новый пакет, а затем удаляет старый пакет, таким образом, файлы от старого пакета не будут мёртвым грузом висеть в ваших каталогах. Если имя обновляемого пакета изменилось, воспользуйтесь вторым вариантом обращения к программе, указав имя старого пакета (который установлен в системе) и после знака процента имя нового пакета (который вы устанавливаете).

Если вы указали переменную окружения ROOT, до выполнения removepkg, то путь, указанный в переменной будет использован вместо корневого каталога. Это полезно для настройки новых дисков для вашего корневого каталога. Они обычно будут смонтированы к /mnt или что-то другое, отличное от /.

upgradepkg не безупречна. Вам всегда следует создавать копии ваших конфигурационных файлов. Если они будут переписаны, у вас всё ещё будет копия оригиналов.

Как и с installpkg и removepkg, вы можете указать несколько пакетов, или использовать шаблоны для имён пакетов.

rpm2tgz/rpm2targz

Red Hat Package Manager это популярный сегодня тип системы пакетов. Многие производители программ предоставляют свои программы в RPM формате. Так как это не наш родной формат, мы не рекомендуем нашим пользователям использовать их. Тем не менее, некоторые вещи доступны только в RPM (даже исходники).

Мы предоставляем программы, которая конвертирует RPM пакеты в наш родной .tgz формат. Это позволит вам распаковать пакет (скажем, при помощи explodepkg ) в временный каталог и исследовать содержимое.

Программа rpm2tgz создаст Slackware пакет с .tgz расширением, а rpm2targz создаст архив с .tar.gz расширением.

**Создание пакетов**

Создание Slackware пакетов может быть, как простым, так и сложным. Нет специфического метода создания пакетов. Единственное требование это то, что пакет должен быть tar gziped файл, и если нужен после-установочный скрипт, он должен быть в каталоге /install/doinst.sh.

Если вы заинтересованы в создании пакетов для вашей системы, или для сети, которую вы обслуживаете, вам следует посмотреть на различные build скрипты в дереве каталогов исходников Slackware. Есть несколько методов, которые мы используем для создания пакетов.

**explodepkg**

explodepkg (8) проделает то же, что installpkg делает при распаковке пакета, но на самом деле не установит сам пакет, и не произведёт никаких изменений в базе данных пакетов. Она просто развернёт содержимое пакета в текущий каталог.

Если вы посмотрите На дерево каталогов исходников Slackware, вы увидите, как мы используем эту команду для ''структурных'' пакетов. Эти пакеты содержат скелет того, как конечный пакет будет выглядеть. Они хранят все необходимые имена файлов (нулевой длины), права доступа и принадлежности. build скрипт проделает cat с содержанием пакета из исходного каталога в build каталог пакета.

**makepkg**

makepkg(8) запакует текущий каталог в формат Slackware package. Она найдёт все символьные ссылки в дереве и добавит блоки их создания в пост-установочный скрипт пакета. Так же программа выдаст сообщение, если обнаружит файлы нулевой длины в дереве пакета.

Эта программа обычно выполняется после того, как вы создали дерево вашего пакета.

**Создание tags и tagfiles (для программы установки)**

Программа установки Slackware производит установку пакетов программ в вашу систему. Существуют так же файлы, которые говорят программе установки, какие из файлов должны быть установлены, какие из них необязательны, и какие по умолчанию должны быть выбраны программой установки.

**tagfile** находится в каталоге первой серии программ и называется tagfile. Он перечисляет пакеты различных дисков и их статус. Статус может быть:

|  |  |
| --- | --- |
| **Опция** | **Действие** |
| ADD | Пакет необходим для нормальной работы системы |
| SKP | Пакет будет автоматически пропущен |
| REC | Пакет не является необходимым, но рекомендован |
| OPT | Пакет необязателен |

Формат прост:

<package name>: <status>

Один пакет на строчку. Оригинальные tagfiles для каждого раздела программ хранятся в tagfile.org. Так что если вы запутались со своими, вы можете восстановить оригинальные.

Многие администраторы предпочитают писать свои собственные tagfiles и затем при установке просто нажимают ''full''. Программа установки прочтёт tagfiles и произведёт установку в соответствии с их содержанием.Если вы используете REC или OPT, то появится окно диалога, которое спросит, следует ли устанавливать этот отдельный пакет. Поэтому рекомендуется использовать ADD и SKP, если вы пишите tagfiles для автоматизированной установки.

Просто убедитесь, что ваши tagfiles будут расположены в тех же местах, что и оригинальные. Или вы можете так же выбрать ''указать специфический путь к пользовательскому tagfile''.

**Практическая часть**

1. Работа с редактором vi
   * Открытие файла
   * Создание
   * Изменение
   * Запись
   * Перемещение по файлу и поиск
   * Выход без записи
   * Использование методов копирования и вставки
2. Tar
   * Создать тарбол из файлов
   * Создать тарбол из каталога с файлами и подкаталогами
   * Добавить к предыдущему тарболу файл и/или каталог
   * Создать тарбол с архивацией (gz, bz2 и пр.)
   * Распаковать архив и записать список извлечённых файлов в файл.
3. Изучить возможности **pkgtool**