Создание пакетов Packman

Содержание

1. Создание пакета Packman
    1. Создание пакета MINIX 3
    2. Отправка пакета MINIX 3

Создание пакета MINIX 3

Мы активно поощряем людей писать и портировать программное обеспечение на MINIX 3. MINIX3 пакеты имеют стандартный формат, который требуется менеджеру пакетов easypack. Вот простые правила. Пожалуйста придерживайтесь их, и они сделают обработку и установку Вашего программного обеспечения для нас и других людей простой и без сбоев.

1. Каждый пакет должен иметь уникальное имя, например "foo-1.2.4", где часть, предшествующая дефису - название программы, и часть после дефиса - номер версии. Вместе они образуют имя пакета.
2. Корневой каталог Вашего пакета должен содержать исполняемый скрипт командной оболочки с именем "build.minix", который собирает и устанавливает пакет. Например:

    ```
    #!/bin/sh
    make
    make install
    ```

    Этот скрипт будет вызываться:
    - **binpackage**, который собирает бинарные пакеты из исходных файлов.
    - **easypack**, который скачивает исходные пакеты, и собирает их них программы.

    Все вызовы будудт осуществляться с опцией *sh* `-e`, так что неудачные команды будудт прерывать сценарий. Для некоторых пакетов возможно потребуются дополнительные (конфигурирующие) команды; но в большинстве случаев пакет должен собираться и устанавливаться правильно выполнив "sh build.minix" (однако см. ниже о .postinstall). Easypack полагается на такое поведение (в старых версиях использовалось только "build"; так что возможно потребуется с Вашей версией easypack попробовать оба названия). Сценарий "build.minix" должен нормально завершаться если пакет установился правильно, иначе завершаться с ошибкой.
3. Не трогайте файлы системы вне Вашего дерева сборки, которые не относятся к пакету (например вызвав *chmem*). Сценарий binpackage будет думать, что это часть Вашего пакета, и включит этот файл.
4. Процедура сборки не должна полагаться на то, что файлы имеют корректный режим доступа (так например исполняемые). Например, не используйте **./configure** в "build.minix"; используйте **sh -e configure**.
5. Устанавливайте PREFIX в "/usr/local"; но посмотрите следующее правило о библиотеках.
6. Библиотеки должны устанавливаться в "/usr/local/lib/ack" или "/usr/local/lib/gcc" или в оба каталога, взависимости от используемых компилятора и компоновщика.
7. Для компиляции программ Вы можете использовать *cc* или *gcc*, хотя мы по

возможности предпочитаем *cc*. Эти два компилятора выдают разные предупреждения, и как правило они оба правы. Для C++ программ используйте *g++*, доступный на веб-сайте Minix.

8. Команда **make install** должна использовать программу *install* для копирования Вашей программы в "/usr/local/bin". Этот метод будет работать даже тогда, когда пользователь вошёл в систему как *bin* (не *root*!).
9. Каждая программа должна иметь *man* страницу в стандартном формате. Команда **make install** должна инсталлировать *man* страницы в соответствующие подкаталоги "/usr/local/man". Большинство *man* страниц о программах помещаются в "/usr/local/man/man1"; но расширенные руководства пользователя должны быть в "/usr/local/man/man9".
10. Сделайте Ваш "build.minix" сценарий таким, чтобы при повторном запуске он устанавливал всё снова. Мало того, что это в значительной степени поможет после неудачной сборки, но наш механизм для создания бинарных пакетов опирается на это.
11. Не включайте в исходники пакета любые сгенерированныме файлы, такие как объектные файлы или сгенерированные Makefiles. Применяйте ваши изменения к уже существующим файлам, а так же путём включения опций или установив переменные окружения в сценарии "build.minix". С добавлением новых файлов всё впорядке.
12. *(Здесь будет правило о дополнительном сценарии ".postinstall".)*
13. Добавьте "**.**descr" файл в корневой каталог пакета с коротким (до 50 символов) однострочным описанием.
14. Аналогичным образом, добавьте для этого пакета список зависимостей сборки ("**.**bdeps") и список зависимостей времени выполнения (runtime) ("**.**rdeps"), оба в формате обычного текстового списка имён пакетов (включая номера версий) по одному в каждой строке.
15. Для создания пакета выполните:

    ```
    cd ..; tar cf - foo-1.2.4 | bzip2 >foo-1.2.4.tar.bz2
    ```

16. Вы можете проверить может ли Ваш исходный пакет собрать корректный бинарный пакет выполнив

    ```
    binpackage yourpackagedir packagesdir
    ```

    Binpackage запустит скрипт сборки и после этого будет сканировать всю Вашу файловую систему (за вычетом ряда каталогов, которые не должны содержать новые файлы пакета, но ожидают изменённые файлы) в поисках обновлений, которые затем, как ожидается, станут частями установленного пакета. Полученный пакет бутет в "*packagesdir*/*yourpackagedir*.tar.bz2". Вот почему Ваш сценарий сборки не должен трогать любые файлы, расположенные вне Вашего пакета.

Отправка пакета MINIX 3

После тщательного тестирования пожалуйста поместите Ваш пакет на трэкер "New Packages", создав там новую тему. Вместо загрузки больших файлов Вам предлагается представить Ваш пакет в виде двух частей:

- Ссылки на исходные, немодифицированные исходные коды программного обеспечения.
- Вашего патча (изменений) на основе этих немодифицированных исходных кодов.

Патч должен включать все Ваши изменения для MINIX (включая сценарий build.minix, .descr, .rdeps, .bdeps и т.п.). Патч должен быть в унифицированном diff формате, т.е.

установите пакет GNU diff и используйте что-то подобное:

- `/usr/local/bin/diff -urN original_src_dir modified_src_dir`

Тщательно ознакомьтесь с патчем перед отправкой. Пожалуйста, проверьте, что ваш патч не включает каких-либо посторонних изменений и убедитесь, что он применяется в отношении чистого, оригинального и неизмененного исходного пакета.

Трэкер "New Packages" находится здесь: New Packages Tracker.

Вы можете также разместить сообщение в группе новостей Google. Пожалуйста не посылайте любые большие файлы в эту группу. Автор должен быть доступен для обратной связи.

Имейте в виду, что рассмотрение новых пакетов (в настоящее время) это трудоемкий процесс, занимающий некоторое время, прежде чем мы примем к рассмотрению Ваш пакет. Спасибо!

MinixWiki: Руководство разработчика/создание пакетов Packman (изменено 2009-11-03 22:50:18 by ArunThomas)

Руководство по портированию

Введение в руководство по портированию

Портирование программного обеспечения в MINIX из Linux или FreeBSD является во многих случаях нетривиальным, но и не заоблачным. Главная часть работы часто заключается в изменении/добавлении makefiles или скриптов сборки. Перекодирование больших частей программного обеспечения требуется только для программ, которые частично или полностью выполняются в пространстве ядра. Вы можете помочь MINIX портируя пакеты!

Для начала прочитайте Руководство разработчика/Posix и Minix, Руководство разработчика/Программирование в Minix и Руководство разработчика/Создание пакетов Packman. Информация этих руководств несколько избыточна.

Портирование Posix приложений в Minix3

Эта страница будет служить отправной точкой для разработчиков, желающих портировать POSIX приложения (например из Linux) в MINIX 3.

Множество приложений уже портировано благодаря усилиям наших авторов. Проверь наличие портированного пакета в репозитариях MINIX 3 прежде чем что-либо портировать, возможно Ваш любимый пакет уже портирован.

Если нет, то поступайте следующим образом:

1. Напишите в группу новостей MINIX и проинформируйте участников о Вашем намерении. Возможно кто-то уже работает над интересующим Вас пакетом.
2. Скачайте исходные коды интересующего Вас приложения.
3. Измените исходные коды чтобы скомпилировать и запустить приложение на MINIX 3.
4. Уведомите разработчиков MINIX 3 о Вашем успехе. Уведомите основного разработчика пакета, потому что он возможно хотел бы включить Ваши изменения в основную версию.
5. Для включения пакета присылайте его на веб-сайт и/или в официальный репозиторий.

Рекомендации для создателей пакета

- Добавьте *.descr* файл, содержищий однострочное описание пакета. (Когда *packman* показывает список пакетов, он отображает это краткое описание пакета с правой стороны. Оно берётся из *.descr* файла пакета. В списке пакетов есть хорошие примеры того, что должно быть в этом файле).

- Добавьте *readme.txt_Minix3* или аналогичный файл, описывающий Ваши изменения. Вы также должны включить Вашу контактную информацию. (Смотрите другие пакеты).

- Добавьте *build.minix* скрипт командной оболочки. (Смотрите другие пакеты).

- Не упаковывайте сгенерированные файлы.

Различные заметки для существующих портов

Некоторые пакеты были портируемы изначально так, что не требовалось изменения исходных кодов чтобы скомпилировать их под MINIX 3. Вы можете использовать эти пакеты как ссылку (примеры) для практики портируемого программирования:

```
CSSC ascii atk automake avra bc bchunk bcrypt btyacc
catdoc cpio ctags diffutils libiconv libmcrypt libpng
links lzo lzop m4 mdf2iso nano nasm neon nomarch
pkg prng pstotext psutils readline rman sed slang slrn
src2tex wdiff webcpp whichman zlib
```

Рекомендации для портирующих

Конфигурация

Сначала выполните *configure* скрипт. Некоторые скрипты используют возможности, не поддерживаемые командной оболочкой ASH (по умолчанию в MINIX). В этом случае используйте другую командную оболочку, такую как BASH. Таким же образом *make*, поставляемый с MINIX, тоже имеет некоторые ограничения. Если он выдаёт какие-либо сообщения об ошибках, попробуйте использовать вместо него GNU Make (*gmake*).

По крайней мере, Вам возможно понадобится определение строки *uname*, возвращаемой MINIX. Это может быть сделано так же как в других POSIX системах, таких как Linux и BSD.

Компилирование

Попробуйте скомпилировать программу. Если Вы столкнулись со многими неожиданными синтаксическими ошибками, то исходный код может основываться на C99 возможностях или GCC расширениях языка C. Один типичный пример - это `//` однострочные комментарии вместо `/*` ... `*/` комментариев. Однострочные комментарии разрешены C99; но ACK к несчастью не реализованы в стандарте C99. В таком случае лучше компилировать программу компилятором GCC.

Во-первых, не реализовывайте отсутствующие библиотечные функции, а лишь объявите их в заголовочных файлах. Таким образом, вы сможете увидеть, каких наиболее важных функций нехватает, прежде чем продолжить работу. На основании отсутствующих функций можно определить, действительно ли портирование осуществимо. Например может оказаться невозможным портирование без существенных изменений программ, которые используют нити или отображаемые в памяти файлы. Такое портирование может быть отложено до появления необходимой функциональности в MINIX. В таком случае, Вы можете добавить как порт так и недостающие функции в список пожеланий, указав об их отсутствии.

Линковка

Если вы добавили объявления, как это рекомендовано в предыдущем разделе, в конечном итоге вы получите ошибку компоновщика, показывающую какие символы были использованы, но не определены. Время реализовать эти недостающие библиотечные функции, предпочтительно таким образом, чтобы они легко могли быть перенесены позже в библиотеку MINIX. Man страницы Open Group website являются хорошим источником информации о том, что конкретно должны делать эти вызовы.

Устранение предупреждений

Предупреждения, выводимые компилятором C, являются хорошим источником информации о том, что может быть неправильно (с точки зрения Minix) в портируемой программе. Желательно устранить все предупреждения, так как это поможет сохранить Вам много времени в процессе тестирования. Несмотря на то, что *cc* обеспечивает некоторые предупреждения, попытайтесь скомпилировать Вашу программу с помощью *gcc* с `-Wall -W` флагами для максимальной помощи со стороны компилятора.

Тестирование

Тестирование и устранение ошибок может потребовать значительного времени для программного обеспечения с плохой переносимостью или использующего функции, отсутствующие в MINIX. Часто с программой идут тестовые наборы, которые обеспечивают хорошую отправную точку.

Различные заметки

- gnutools находятся в "/usr/gnu/bin"; Вам может понадобиться PATH в Makefiles.
- Добавьте "README_Minix.txt" файл, описывающий Ваши изменения, в соответствии с особенностями MINIX 3.
- Используйте *gar*, вместо *ar*, когда создаёте GCC библиотеки.

- Если Вы устанавливаете библиотеки для использования другими программами, то поместите библиотеки, собранные с помощью *cc*, в */usr/local/lib/ack/*; и поместите библиотеки, собранные с помощью *gcc*, в */usr/local/lib/gcc/*.

- !!!!!!!Не комментируйте код -- используйте директивы препроцессора!!!!(Прим не уверен не понимаю). **`__minix`** символ препроцессора указывает, что C код компилировался на MINIX.

- В некоторых Makefiles, "`-funsigned-char`" добавлен в CFLAGS.

- Битовые поля не распознаются нативно *cc* программами. Вы можете использовать GCC компилятор или удалить битовые поля из структур -- это безвредно, если данные структуры не должны строго соответствовать структуре оборудования или памяти.

- Постарайтесь, чтобы ваши патчи были максимально ненавязчивыми, чтобы повысить шансы на включение их в основной пакет.
- Пожалуйста построчно комментируйте Ваши изменения.
- **ioctl** and **setsockopt** являются источниками проблемы портирования. Вызовы функций могут быть определены, однако не иметь, или иметь ограниченную реализацию, так что Вы заранее можете проверить все эти вызовы. В частности, убедитесь что каждый вызов должным образом проверен на возвращаемое сообщение об ошибке.

- В MINIX вызовы **recv()** и **send()** не поддерживают флаги.

- Нет способа запустить сервер, слушающий на *localhost* (127.0.0.1), потому что нет выделенного loopback устройства. Если Вы столкнулись с этой проблемой, можете взамен использовать прослушивание на **INADDR_ANY**, но отметьте, что это работает только при существующем подключении к сети (например X Window System не может быть запущена без него). Также подумайте о проблеммах безопасности, связанных с принятием нелокальных сетевых соединений.

- Используйте последнюю версию MINIX. Это делает пакет более полезным, так как обеспечивает Вас актуальным функционалом.
- Если Вы используете GCC, становится возможным применять в MINIX 64-bit **long long** тип. Тем не менее это не поддерживается библиотечными функциями и может быть обрезано когда используется. Проверяйте в этом случае значения большие 2^{32}.

MinixWiki: Руководство разработчика/Руководство по портированию (последние изменения 2009-11-03 21:48:48 David van Moolenbroek)