УДК 004.725.5

Стрельцов Д.С
Бутенко Е.А
студенты специалитета
Российский государственный университет нефти и газа (НИУ) имени И.М Губкина, г.Москва

ОСОБЕННОСТИ СБОРКИ RPM ПАКЕТА ДЛЯ ALT LINUX, СРАВНЕНИЕ С REDHAT

Аннотация: в данной статье рассмотрим особенности настройки RPM-пакета в виртуальной среде VirtualBox при помощи виртуальной машины с операционной системой Alt Linux, а также сравним с операционной системой Redhat.

Ключевые слова: пакет PRM, сборка пакета PRM, особенности сборки.

UDC 004.725.5

Streltsov D. S.
Butenko E.A.
students of the specialty
Gubkin Russian State University of Oil and Gas, Moscow

PECULIARITIES OF THE ALT LINUX RPM PACKAGE BUILD AND COMPARISON WITH REDHAT

Abstract: in this article, we will consider the features of setting up an RPM package in the VirtualBox virtual environment using a virtual machine with the Alt Linux operating system and also compare it with the Redhat operating system.

Key words: PRM package, PRM package assembly, assembly features.

Введение

Система управления пакетами играет ключевую роль в администрировании Linux-дистрибутивов, обеспечивая удобную работу с ПО. Одним из наиболее распространенных форматов пакетов Linux является RPM (Red Hat Package Manager), который активно используется в таких дистрибутивах, как Alt и Red Hat Linux. Однако каждый дистрибутив может иметь свои особенности в сборке и управлении RPM-пакетами.

В данной статье мы рассмотрим особенности сборки RPM-пакетов для дистрибутива Альт Линукс. Мы сравним процессы сборки с аналогичными процессами в Red Hat.

На просторах интернета достаточно информации о сборки пакетов RPM на разных Linux-дистрибутивов, но в нашей статье мы расскажем, как собрать RPM пакет на Альт Linux, исправим частую проблему при создании пакета, расскажем про то, как отличается структура в Альт Linux от Red Hat.

Объект исследования — это область операционных систем, основанных на ядре Linux, а также процессы и инструменты, связанные с разработкой, сборкой и распространением программных пакетов RPM для Red Hat и Альт Линукс.

Предмет исследования — это особенности сборки пакетов RPM для Red Hat и Альт Линукс.

Цель исследования — собрать пакет RPM для Альт Линукс, провести сравнительный анализ с Red Hat, выявить ключевые различия и сходства.

RPM Package Manager

RPM или RPM Package Manager - это пакетный менеджер, используемый в дистрибутивах Linux, основанных на Red Hat. Такое же название имеет формат файлов этого пакетного менеджера. RPM предоставляет возможность управления сотнями и тысячами отдельных пакетов. Каждый пакет - дискретный набор компонентов одного приложения или библиотеки (файлы приложения, документация, файлы конфигурации).

Практически каждый крупный проект, использующий RPM, имеет свою версию пакетного менеджера, отличающуюся от остальных. Между представителями семейства RPM могут иметься следующие различия:

1. наборы макросов, используемых в spec-файлах;
2. различное поведение RPM при сборке «по умолчанию» – при отсутствии каких-либо указаний в spec-файлах;
3. формат строк зависимостей;
4. мелкие отличия в семантике операций (например, в операциях сравнения версий пакетов);
5. мелкие отличия в формате файлов.

Для пользователя различия чаще всего заключаются в невозможности поставить «неродной» пакет из-за проблем с зависимостями или из-за формата пакета.

RPM в ОС Альт

Сизиф (Sisyphus) — проект по разработке репозитория (хранилища) RPM-пакетов, входит в 10 крупнейших в мире банков пакетов свободных программ (в 2007 году был 5-м).

В рамках проекта Сизиф членами ALT Linux Team разработан набор инструментов:

1. **Hasher** — инструмент для безопасной сборки RPM-пакетов в контролируемой среде. Hasher – инструмент для сборки пакетов в «чистой» и контролируемой среде. Это достигается с помощью создания в

chroot минимальной сборочной среды, установки туда указанных в source-пакете сборочных зависимостей и сборке пакета в свежесозданной среде. Для сборки каждого пакета сборочная среда создается заново.

Такой принцип сборки имеет несколько следствий:

- все необходимые для сборки зависимости должны быть указаны в пакете. Для облегчения поддержания сборочных зависимостей в актуальном состоянии в Sisyphus используется инструмент buildreq;
- сборка не зависит от конфигурации компьютера пользователя, собирающего пакет, и может быть повторена на другом компьютере;
- изолированность среды сборки позволяет с легкостью собирать на одном компьютере пакеты для разных дистрибутивов и веток репозитория – для этого достаточно лишь направить hasher на различные репозитории для каждого сборочного окружения.

2. **Gear** (Get Every Archive from git package Repository) – система для работы с произвольными архивами программ. В качестве хранилища данных gear использует git, что позволяет работать с полной историей проекта.

Идея gear заключается в том, чтобы с помощью одного файла с простыми правилами (для обработки которых достаточно sed и git) можно было бы собирать пакеты из произвольно устроенного git-репозитория, по аналогии с hasher, который был задуман как средство для сборки пакетов из произвольных «srpm-пакетов».

Gear поддерживает полный цикл организации репозиториев: создание репозитория или импорт существующих src.rpm-пакетов, обновление upstream-кода в репозиториях, наложение патчей и пакетирование, экспорт pkg.tar и src.rpm, сборка бинарных RPM-пакетов.

3. Alterator — платформа для управления конфигурацией Linux-системы.
4. ALT Linux Installer — инсталлятор, используемый в дистрибутивах ALT Linux.

В репозитории собираются пакеты для следующих архитектур: x86_64, aarch64, ppc64le, i586, armhf, e2k.

Основные отличия RPM в «Альт» и Сизиф от RPM других крупных проектов:

1. обширный набор макросов для сборки различных типов пакетов;
2. отличающееся поведение «по умолчанию» для уменьшения количества шаблонного кода в spec-файлах;
3. наличие механизмов для автоматического поиска межпакетных зависимостей;
4. наличие так называемых set-version зависимостей (начиная с 4.0.4-alt98.46), обеспечивающих дополнительный контроль над изменением ABI-библиотек;
5. до p8 и выпусков 8.x включительно – очень древняя версия «базового» RPM (4.0.4), от которого началось развитие ветки RPM в Sisyphus (в Sisyphus и p9 осуществлен частичный переход на rpm 4.13).

Различают два вида пакетов RPM:

1. пакет с исходным кодом – SRPM-пакет (расширение .src.rpm). Пакет src.rpm можно использовать только для сборки двоичных пакетов, но не установки.
2. собранный двоичный пакет – RPM-пакет (расширение вида .rpm).

Установка rpm пакета с помощью rpm-build на альт linux.

Для примера сборки пакета будем использовать программу для вывода строки. Ссылка на gitlab-репозиторий с исходными текстом программы на языке C++ (https://gitlab.basealt.space/alt/edu/ExampleFirstProject.git)

1. Заходим под администратором и делаем обновление.

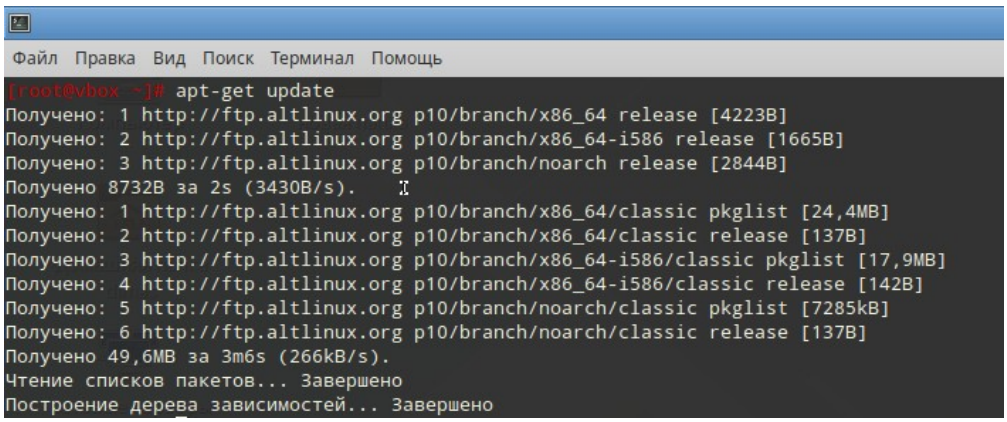


Рис. 1. Apt-get update

2. Устанавливаем необходимые инструменты для сборки пакетов с помощью rpmbuild.

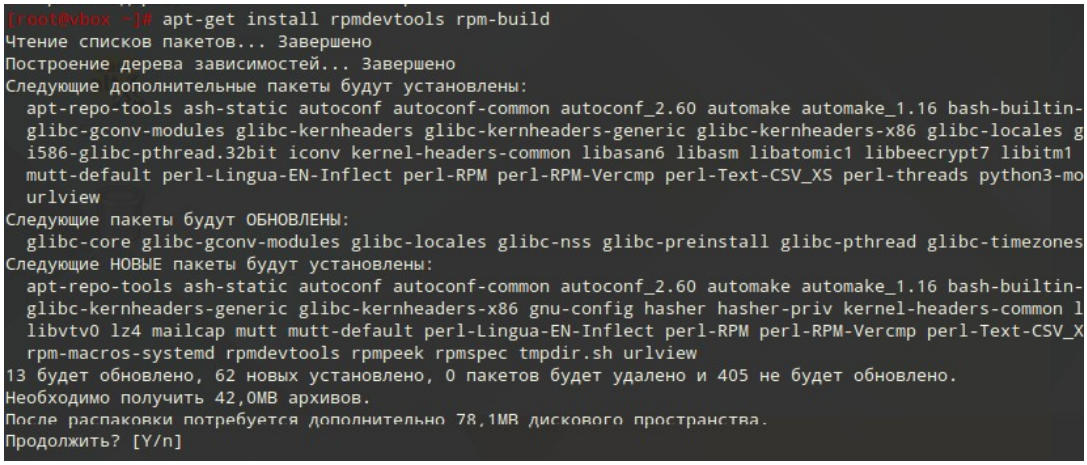


Рис. 2. Установка rpmdevtools rpm-build

Вводим y-yes и продолжаем установку.

Рис. 3. Установка rpmdevtools rpm-build

3. Также прописываем команду, которая устанавливается для компиляции программ на языках C и C++, так как наш файл будет на C++.

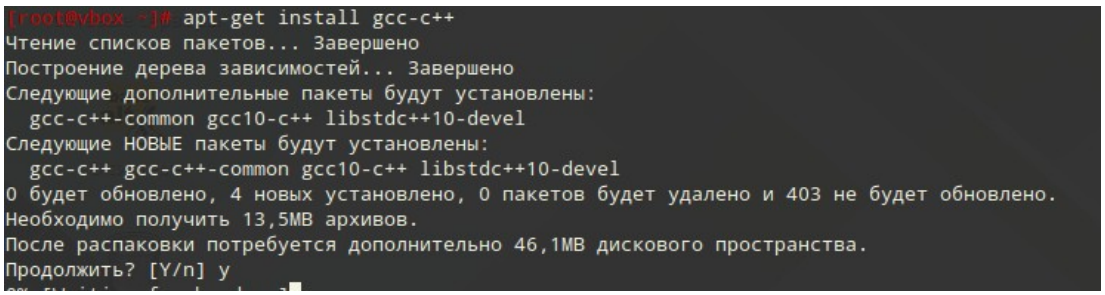


Рис. 4. Установка gcc-c++

Выбираем y-yes, продолжаем установку.

Рис. 5. Установка gcc-c++

4. Выходим из администратора и далее работаем под пользователем.

Рис. 6. Выход из root

5. Прописываем команду **rpmdev-setuptree**, которая нужна для создания структуры каталогов (дерева) для сборки RPM-пакетов. Далее просматриваем каталоги. RPM использует для сборки пакетов пять каталогов, они описаны в таблице.

Таблица 1. Структуры каталогов

Каталог	Использование
BUILD	Содержит все файлы, которые появляются при сборке пакета.
RPMS	Здесь формируются собранные RPM-пакеты (.rpm) в подкаталогах для разных архитектур, например, в подкаталогахx86_64 и noarch.
SOURCES	Здесь находятся архивы исходного кода и патчи. Утилита rpmbuild ищет их здесь.
SPECS	В этот каталог помещаются spec-файлы всех rpm-пакетов, которые запланированы на сборку.
SRPMS	Утилита rpmbuild помещает в этот каталог собранные src.rpm-пакеты с исходным кодом.

Рис. 7. Структура каталогов дерева

6. Клонируем репозиторий. Это даёт нам доступ ко всем веткам в этом репозитории, и вы можете легко переключаться между ними, чтобы просматривать каждую версию и её файлы.

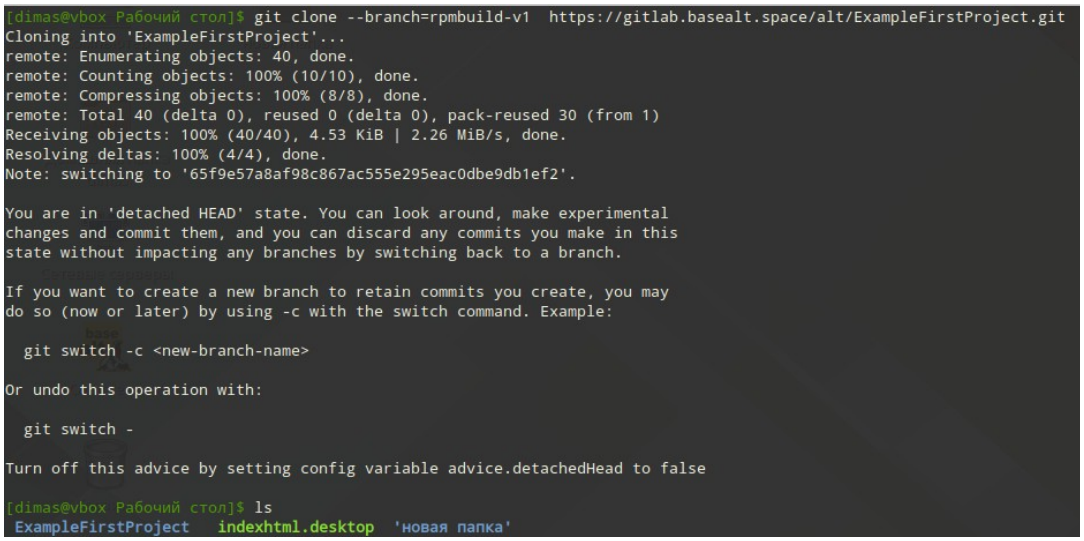


Рис. 8. Клонирование репозитория в папку

7. Переходим в наш проект ExampleFirstProject.

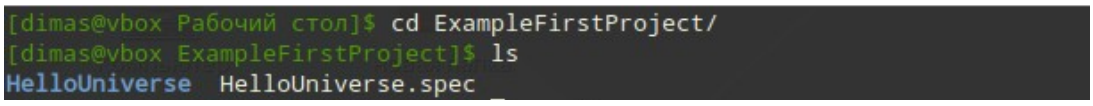
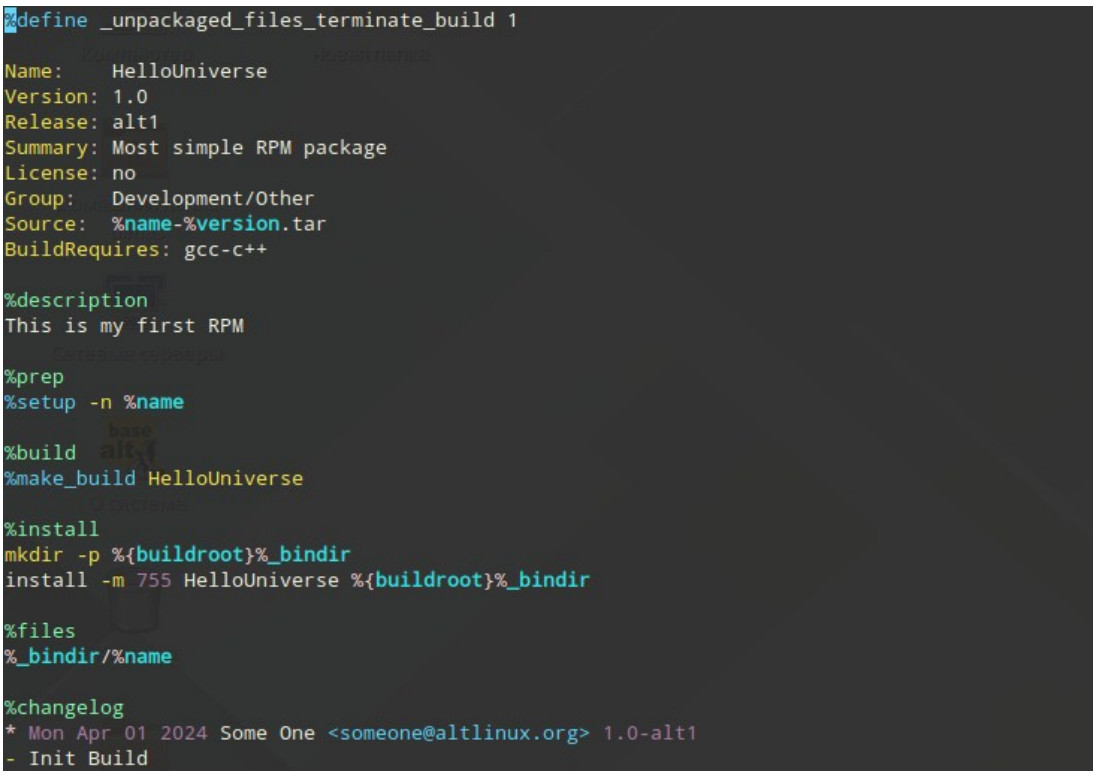


Рис. 9. Переход в папку ExampleFirstProject

8. Просмотр конфигурации spec - файла осуществляется с помощью команды vim HelloUniverse.spec.



```
%define _unpackaged_files_terminate_build 1

Name:    HelloUniverse
Version: 1.0
Release: alt1
Summary: Most simple RPM package
License: no
Group:   Development/Other
Source:  %name-%version.tar
BuildRequires: gcc-c++

%description
This is my first RPM

%prep
%setup -n %name

%build
%make_build HelloUniverse

%install
mkdir -p %{buildroot}%_bindir
install -m 755 HelloUniverse %{buildroot}%_bindir

%files
%_bindir/%name

%changelog
* Mon Apr 01 2024 Some One <someone@altlinux.org> 1.0-alt1
- Init Build
```

Рис. 10. Конфигурация HelloUniverse.spec

Spec-файл можно рассматривать как «инструкцию», которую утилита rpmbuild использует для фактической сборки RPM-пакета. Spec-файл определяет все действия, которые должны быть выполнены при сборке RPM-пакета, а также все действия, необходимые при установке/удалении пакета. Каждый src.rpm-пакет имеет в своем составе spec-файл.

Особенная для ОС Альт директива преамбулы spec-файла является Release - релиз пакета используется для указания номера сборки пакета при данной версии upstream-кода. Для пакетов Sisyphus поле Release должно иметь вид в простых случаях – altN, а в сложных – altN[суффикс]. N начинается с 1 для каждой новой upstream-версии и увеличивается на 1 для каждой новой сборки: 1.0-alt1, 1.0-alt2.

Таблица 3. Директивы основной части spec-файла

SPEC Директива	Определение
%description	Описание ПО, входящего в комплект поставки RPM. Длина каждой

	строки не должна превышать 72 символа. Учитывается при поиске пакета через apt-cache search, выводится во время просмотра информации о пакете при помощи apt-cache show имя_пакета.
%prep	Команда/серия команд для подготовки ПО к сборке, можетсодержать сценарий оболочки (shell скрипт).
%build	Команда/серия команд для фактической сборки программного обеспечения в машинный.
%install	Команды установки/копирования файлов из сборочного каталога в псевдо-корневой каталог. Во время сборки пакета этот раздел эмулирует конечные пути установки файлов в систему.
%check	Команда/серия команд для тестирования ПО.
%files	Список файлов, которые будут установлены в системе конечного пользователя.
%changelog	Запись изменений, произошедших в пакете между сборками разных версий или релизов.

9. Переходим в папку HelloUniverse. После этого заходим в файл HelloUniverse.cpp, с помощью команды: **vim HelloUniverse/ HelloUniverse.cpp,** это файл с функцией на C++, которая будет вызываться в нашем PRM пакете.

```
[dimas@vbox ExampleFirstProject]$ cd  HelloUniverse
[dimas@vbox HelloUniverse]$ ls
HelloUniverse.cpp  Makefile
```

Рис. 11. Переход в директорию

```
#include <iostream>
int main()
{
    std::cout << "I LOVE AU.TEAM <3\n";
    return 0;
}
```

Рис. 12. C++ код в HelloUniverse.cpp

10. Просмотр Makefile, так же аналогично с помощью команды **vim HelloUniverse/ Makefile.** Выходим из HelloUniverse.

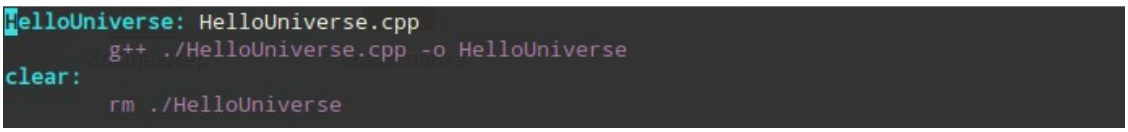


Рис. 13. Конфигурация Makefile

11. Как можем заметить у нас есть файл **HelloUniverse/ HelloUniverse.cpp,** но ещё создался (дублировался) файл **HelloUniverse/ HelloUniverse.cpp~,** от которого нам необходимо избавиться. Просматриваем этот дублированный файл, он совпадает с исходным файлом, удаляем его.

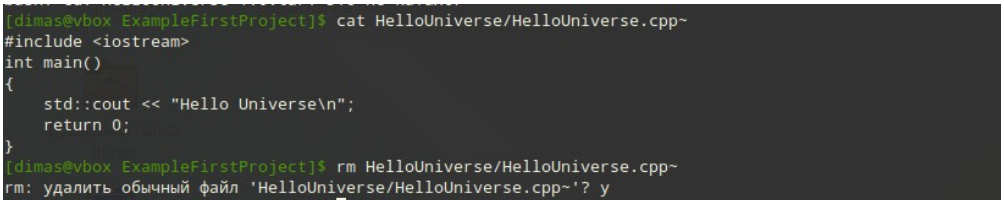


Рис. 14. Конфигурация Makefile

12. Создаём архив из нашей папки с исходными файлами.

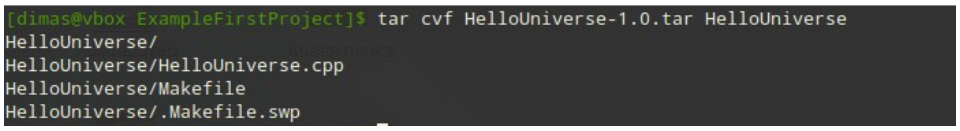


Рис. 15. Конфигурация Makefile

Рис. 16. Содержимое директории

13. Для того, чтобы приступить к сборке пакета осталось выполнить два действия: поместить архив в SOURCE, а spec-файл в SPECS.

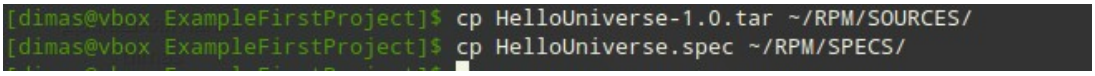


Рис. 17. Копирование архива и spec-файл в директории

14. Переходим к сборке пакета. Опция -b указывает на режим сборки. Второй ключ после -b говорит, до какой стадии осуществлять процесс. –ba - Собрать бинарный пакет и пакет с исходным кодом.

```
[dimas@vbox ExampleFirstProject]$ rpmbuild -ba ~/RPM/SPECS/HelloUniverse.spec
Выполняется(%prep): /bin/sh -e /tmp/.private/dimas/rpm-tmp.95395
+ umask 022
+ /bin/mkdir -p /home/dimas/RPM/BUILD
+ cd /home/dimas/RPM/BUILD
+ cd /home/dimas/RPM/BUILD
+ rm -rf HelloUniverse
+ echo 'Source #0 (HelloUniverse-1.0.tar):'
Source #0 (HelloUniverse-1.0.tar):
+ /bin/tar -xf /home/dimas/RPM/SOURCES/HelloUniverse-1.0.tar
+ cd HelloUniverse
+ /bin/chmod -c -Rf u+rwX,go-w .
+ exit 0
Выполняется(%build): /bin/sh -e /tmp/.private/dimas/rpm-tmp.81005
+ umask 022
+ /bin/mkdir -p /home/dimas/RPM/BUILD
+ cd /home/dimas/RPM/BUILD
+ cd HelloUniverse
+ make -j4 HelloUniverse
make: Entering directory '/home/dimas/RPM/BUILD/HelloUniverse'
g++ ./HelloUniverse.cpp -o HelloUniverse
make: Leaving directory '/home/dimas/RPM/BUILD/HelloUniverse'
+ exit 0
Выполняется(%install): /bin/sh -e /tmp/.private/dimas/rpm-tmp.61403
+ umask 022
+ /bin/mkdir -p /home/dimas/RPM/BUILD
+ cd /home/dimas/RPM/BUILD
+ /bin/chmod -Rf u+rwX -- /tmp/.private/dimas/HelloUniverse-buildroot
+ :
+ /bin/rm -rf -- /tmp/.private/dimas/HelloUniverse-buildroot
+ PATH=/usr/libexec/rpm-build:/home/dimas/bin:/usr/local/bin:/usr/bin:/bin:/usr/games
+ cd HelloUniverse
+ mkdir -p /tmp/.private/dimas/HelloUniverse-buildroot/usr/bin
+ install -m 755 HelloUniverse /tmp/.private/dimas/HelloUniverse-buildroot/usr/bin
+ '[' '%{buildarch}' = noarch ']'
+ QA_CHECK_RPATHS=1
+ case "${QA_CHECK_RPATHS:-}" in
+ /usr/lib/rpm/check-rpaths
/tmp/.private/dimas/rpm-tmp.61403: line 133: /usr/lib/rpm/check-rpaths: No such file or directory
ошибка: Неверный код возврата из /tmp/.private/dimas/rpm-tmp.61403 (%install)


Ошибки сборки пакетов:
    Неверный код возврата из /tmp/.private/dimas/rpm-tmp.61403 (%install)
```

Рис. 18. Сборка RPM пакета

На данном этапе появляется ошибка: Ошибки сборки пакетов: Неверный код возврата из /tmp/.private/(имя пользователя)/rpm-tmp.61403(%install). С ней сталкиваются многие пользователи. Нам необходимо открыть каталог rpmmacros для редактирования и далее закомментировать находящиеся в нем макросы. Ниже приведены шаги решения ошибки.

```
[dimas@vbox ExampleFirstProject]$ vim ~/.rpmmacros
```

Рис. 19. Редактирование макроса

```
%_topdir        %homedir/RPM
#%_tmppath      %homedir/tmp

# %packager     Joe Hacker <joe@email.address>
# %_gpg_name    joe@email.address

%__arch_install_post \
    [ "%{buildarch}" = "noarch" ] || QA_CHECK_RPATHS=1 ; \
    case "${QA_CHECK_RPATHS:-}" in [1yY]*) /usr/lib/rpm/check-rpaths ;; esac \
    /usr/lib/rpm/check-buildroot
```

Рис. 20. Редактирование макроса

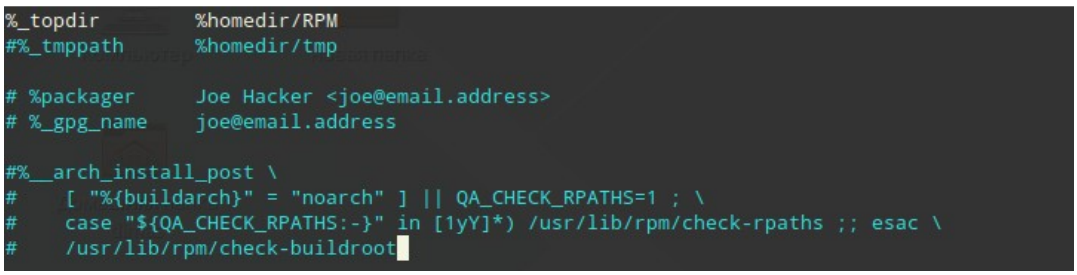


Рис. 21. Редактирование макроса

После, продолжаем сборку пакета.

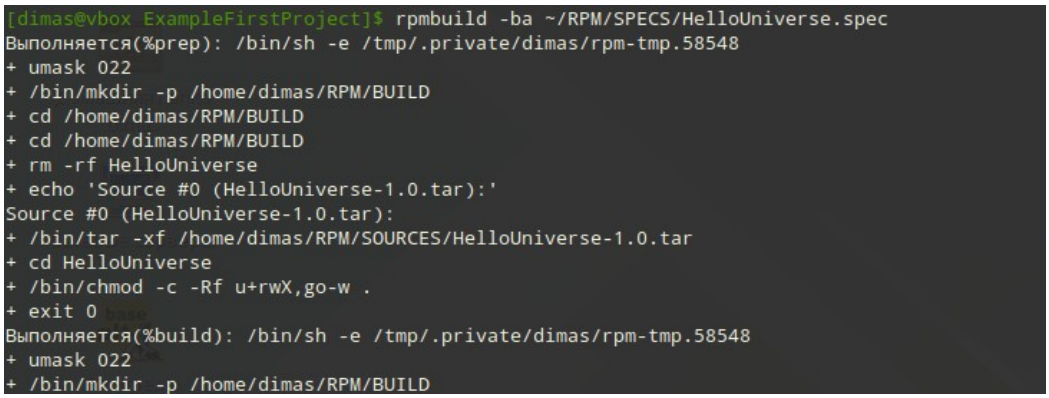


Рис. 22. Сборка RPM пакета

Сборка прошла успешно.

15. Переходим в администратора, устанавливаем пакет RPM. Указываем полный путь к нему.

Рис. 23. Установка RPM пакета через rpm -i

16. Открываем наш пакт **HelloUniverse** и видим, что всё работает.

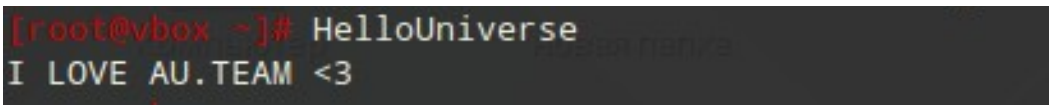


Рис. 24. Вызов RPM пакета

Таким образом, выполнив простые действия в 16 шагов мы собрали RPM-пакет для Альт Linux.

Сравнение с Redhat

Так как формат RPM (Red Hat Package Manager) изначально был разработан для Redhat Linux, что понятно по названию, разница между RPM в Alt Linux и Redhat Linux минимальна, но она всё же есть.

Redhat Linux может использовать Mock и koji. Mock предоставляет поддерживаемое сообществом решение для создания пакетов для различных архитектур и разных версий Fedora или RHEL. Также как и hasher ОС Альт, Mock собирает пакет RPM в чистой среде.

Управление исходниками в Redhat через .spec и SRPM, возможно использование Git вручную, в ОС Альт исходники управляются через gear, интегрированный с Git (каждый пакет хранится в отдельном репозитории).

RPMLint для проверки пакетов. Пакеты тестируются на совместимость с текущими версиями. В ОС Альт Repocop, sisyphus_check. проверяет пакеты по стандартам ALT. Пакеты тестируются и синхронизируются с Sisyphus.

Поддержка x86_64, ARM64, IBM Power, IBM z Systems. В ОС Альт Поддержка x86_64, ARM, Elbrus. использует стандартные RPM-макросы (например, %configure, %makeinstall). ОС Альт поддерживаются те же макросы, но добавляются уникальные ALT-ориентированные макросы.

Таблица 4. Отличия RPM

Аспект	Red Hat Linux	Alt Linux
Структура дерева	/home/{user}/rpmbuild/	/home/{user}/RPM/
Установка пакета	dnf install / rpm -i	apt-get install / rpm -i
Менеджеры пакетов	yum, dnf, rpm	apt, rpm
Сборочная среда	Mock, koji.	Hasher, Gear.
Управление исходниками	Spec и SRPM, возможно использование Git вручную.	Gear, Git (каждый пакет хранится в отдельном репозитории).

Проверка качества пакетов	RPMLint	Repocop, sisyphus_check.
Поддержка архитектур	x86_64, ARM64, IBM Power, IBM z Systems.	x86_64, aarch64, ppc64le, i586, armhf, e2k(Elbrus).
Макросы	Использует стандартные RPM-макросы	Те же макросы, уникальные ALT-ориентированные макросы

Заключение

В данной статье мы научились создавать RPM пакеты на Alt Linux, также в ходе сборки ознакомились с некоторыми особенностями, которые отличают данные процессы от других ОС, таких как Red Hat Linux, также в конце мы провели сравнительный анализ и выяснили, что основные различия касаются структуры каталогов, методов управления зависимостями

Понимание этих процессов позволит разработчикам и системным администраторам более эффективно работать с Alt Linux и использовать его возможности в своих проектах.

Использованные источники:

1. Назначение RPM / [Электронный ресурс] // RedOC : [сайт]. — URL: https://redos.red-soft.ru/base/redos-7_3/7_3-base-consept/7_3-sys-dnf/7_3-manag-pack-rpm/7_3-rpm-view/?nocache=1733514679585 (дата обращения: 06.12.2024).

2. ООО «Базальт СПО» Руководство пользователя / ООО «Базальт СПО» [Электронный ресурс] // BaseAlt : [сайт]. — URL: https://www.basealt.ru/fileadmin/user_upload/manual/ALT_Platform_guide.pdf (дата обращения: 06.12.2024).

3. Технология сборки пакетов RPM / [Электронный ресурс] // ALT Linux Wiki : [сайт]. — URL:

https://www.altlinux.org/Технология_сборки_пакетов_RPM (дата обращения: 06.12.2024).

4. Уймин А.Г. Компьютерные сети. L2-технологии : Практикум/ Уймин А.Г. — Москва : Ай Пи Ар Медиа, 2024. — 191 с. — ISBN 978-5-4497-2539-4. — Текст : электронный.(дата обращения: 06.12.2024)

5. ALT Packaging HOWTO / [Электронный ресурс] // ALT Linux Wiki : [сайт]. — URL: https://www.altlinux.org/ALT_Packaging_HOWTO (дата обращения: 06.12.2024).

6. RPM Packaging Guide / 2024 Red Hat [Электронный ресурс] // Red Hat Documentation : [сайт]. — URL: https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html-single/rpm_packaging_guide/index (дата обращения: 06.12.2024).

7. Valentin Bajrami How to create a Linux RPM package / Valentin Bajrami [Электронный ресурс] // Red Hat : [сайт]. — URL: https://www.redhat.com/en/blog/create-rpm-package (дата обращения: 06.12.2024).