

Часть I. Документы для ALT Linux Team

Глава 1. *ALT* Packaging

ALT specfile conventions

Дмитрий Левин

Преобразование оригинального текста в *DocBook* : Юрий Зотов

<yznews@hotmail.ru>

Обоснование

При разработке этих правил решались следующие задачи:

Обеспечить желаемую функциональность: наши пакеты должны отвечать определённым правилам, о которых пойдёт речь несколько позже. Для этого надо, чтобы срес-файлы обеспечивали выполнение этих правил.

Помочь разработчику: так как срес-файлы все ещё пишут люди, то их работу нужно свести к тому минимуму, который, собственно, и требует участия человека. Разработчик не должен копировать блоки кода из файла в файл, ибо эта неинтеллектуальная работа отнимает массу сил и чревата ошибками. Для этого есть макросы. Если какой-то код появляется в разных срес-файлах более одного раза, то надо написать макрос(ы).

Срес-файлы

Устаревшие конструкции

Не следует использовать устаревшие конструкции — они лишь загромождают срес-файл, снижая тем самым его читабельность. К устаревшим конструкциям, в частности, относятся:

- тэг `BuildRoot`;
- строки вида `rm -rf $RPM_BUILD_ROOT`;
- `%_defattr` со стандартными аргументами в начале файлов и секций `%files`;
- секция `%clean`, пустая либо без разумного содержания.

Фигурные скобки

Нет смысла засорять текст `spec`-файла ненужными фигурными скобками. Избавиться от них можно с помощью команды `cleanup_spec spec-файл`

Выравнивание

Используйте табуляции для выравнивания. Избегайте пробельных символов в конце строк.

Порядок тэгов

Рекомендуемый порядок заголовочных тэгов: Name, Version, Release, Serial, далее Summary, License, Group, Url, Packager, BuildArch, потом Source*, Patch*, далее PreReqs, Requires, Provides, Conflicts, и, наконец, Prefix, BuildPreReqs, BuildRequires. Разумеется, не все из вышеперечисленных тэгов используются, равно как встречаются и другие редко используемые тэги. В связи с тем, что BuildRequires зарезервирован для автоматически вычисляемых зависимостей, для указания особых зависимостей следует использовать BuildPreReq.

Значения тэгов

Значение тэга от его имени следует разделять одним пробелом. Элементы списка значений следует разделять запятой с последующим пробелом. Значение тэга Summary следует начинать с прописной буквы. Значение тэга Summary не следует завершать точкой. Значения тэгов Summary и %description могут содержать названия команд только в не изменённом виде.

Группы

Значение тэга Group должно соответствовать действительности и при этом принадлежать фиксированному множеству, перечисленному в файле `/usr/lib/rpm/GROUPS`.

ChangeLog

При формировании первой строки changelog-записи используйте утилиту `add_changelog spec-файл`. Описание изменений должно быть информативным; недостаточно объявить о наличии изменений, необходимо их все явно перечислить.

Файлы локализации

Если в состав пакета входят файлы локализации либо другие файлы на разных языках, следует использовать макрос `%find_lang`. Подробную информацию можно получить, выполнив команду `/usr/lib/rpm/find-lang -h`.

Внутрипакетные зависимости

При работе с мультипакетными `src`-файлами соблюдайте правило внутрипакетных зависимостей: Если один пакет в какой-либо мере зависит от другого подпакета, то эта зависимость должна быть указана полностью, включая не только имя, но также версию, релиз и `serial` (если есть). Например, «Requires: `%name = %version-%release`» или «Requires: `%name = %serial:%version-%release`». Обратите внимание на синтаксис: знак равенства, в отличие от дефиса, окружён пробелами.

Разделяемые библиотеки

Пакеты, содержащие как разделяемые библиотеки, так и использующие их программы, должны быть разделены на подпакеты таким образом, чтобы в подпакет, содержащий разделяемые библиотеки, не входили использующие их программы. Это, в частности, позволяет уменьшить количество циклических зависимостей. По традиции, имена пакетов, состоящих только из разделяемых библиотек, должны начинаться с префикса `lib` либо содержать его внутри слова. При разделении подпакетов следует помнить о внутрипакетных зависимостях.

Каждый пакет, содержащий разделяемые библиотеки в каталоге `/lib`, `/usr/lib` или `/usr/X11R6/lib`, должен их регистрировать при установке/обновлениях и удалении с помощью макросов `%post_ldconfig` и `%postun_ldconfig` соответственно.

Статические библиотеки

Статические библиотеки должны паковаться в отдельные подпакеты, что связано со спецификой их использования. Если имя `devel`-подпакета заканчивается суффиксом `-devel`, то имя нового `devel-static`-подпакета будет заканчиваться суффиксом `-devel-static`. При разделении подпакетов следует помнить о внутрипакетных зависимостях: в списке зависимостей `devel-static`-подпакета должна присутствовать зависимость от `-devel = %version-%release`.

Переименование пакетов

Иногда пакеты переименовывают. Например, это случается при упаковке разделяемых библиотек. В таких случаях следует указывать правильную информацию о зависимостях, необходимую для корректного обновления. В частности, должен присутствовать:

- тэг `Provides`: `старое_имя = %version`;
- тэг `Obsoletes`: `старое_имя`.

Патчи

Наименование патчей

При создании новых патчей, а также при импортировании патчей из других источников необходимо придерживаться единых правил наименования имён патч-файлов: *NAME-VERSION-ORIGIN-WHAT*.patch, где

- *NAME* и *VERSION* — имя и версия пакета, для которого сделан патч;
- *ORIGIN* — аббревиатуры источников патча (обычно дистрибутивов);
- *WHAT* — краткое описание патча.

В случае, когда патч образован из нескольких частей, полученных из разных источников, компонента имени *ORIGIN* должна содержать аббревиатуры всех источников. Если патч был создан или адаптирован для *ALT Linux*, то в *ORIGIN*, соответственно, должно присутствовать *-alt-*. Для патчей, созданных на базе CVS, компонента имени *ORIGIN* должна начинаться с *cvs-YYYYMMDD*.

При составлении описания патча следует иметь в виду следующие общепринятые сокращения:

makefile

патчи, затрагивающие исключительно **makefile***;

bound

проверки на границы (буфера, целых чисел, и т.п.);

config

патчи, затрагивающие исключительно конфигурационные файлы;

configure

патчи, затрагивающие исключительно **configure***;

doc

патчи, затрагивающие исключительно документацию;

fixes

кумулятивные патчи и/или исправления по надёжности и/или безопасности;

format

патчи на использование форматирования строк (**printf**);

install

патчи, направленные на возможность выполнения **make install** непривилегированным пользователем;

linux

патчи, предназначенные для портирования ПО на Linux;

man

патчи, затрагивающие исключительно man-страницы;

texinfo

патчи, затрагивающие исключительно документацию в формате texinfo;

tmp

патчи, предназначенные для решения различных вопросов, связанных с временными файлами;

`vitmp`

патчи, направленные на поддержку `vitmp(1)`;

`warnings`

патчи, исправляющие ошибки, найденные компилятором.

Исходный код

Формат хранения

Исходный код большого объёма (как архивы, так и патчи, по статусу приравненные к ним) следует хранить в упакованном виде. Метод сжатия, `gzip` или `bzip2`, следует выбирать таким образом, чтобы размер архива был минимальным, например, с помощью утилиты `zme`. Исключение составляют `posource`-пакеты: в них отсутствующие файлы следует указывать в виде корректных URL.

ALT Linux RPM: особенности версии `rpm-4.0.4-alt5`

Дмитрий Левин

Преобразование оригинального текста в *DocBook* : Юрий Зотов

<yznews@hotmail.ru>

Обоснование

При разработке изменений и дополнений к RPM решались следующие задачи:

Обеспечить желаемую функциональность: наши пакеты должны отвечать определенным правилам, о которых пойдёт речь несколько позже. Для этого надо, чтобы `src`-файлы обеспечивали выполнение этих правил.

Помочь разработчику: так как `src`-файлы все ещё пишут люди, то их работу нужно свести к тому минимуму, который, собственно, и требует участия человека. Разработчик не должен копировать блоки кода из файла в файл, ибо эта неинтеллектуальная работа отнимает массу сил и чревата ошибками. Для этого есть макросы. Если какой-то код появляется в разных `src`-файлах более одного раза, то надо написать макрос(ы).

Новые тэги

BuildHost

С помощью этого тэга можно переопределить значение `hostname`, которое RPM помещает в заголовок каждого пакета. По умолчанию, как и ранее, используется значение, возвращаемое `uname(2)`.

Устаревшие тэги

BuildRoot

Времена, когда тэг `BuildRoot` в спес-файле определял, какой каталог RPM будет использовать в качестве `BuildRoot`, прошли безвозвратно. Теперь этот тэг не несёт никакой информации и может (и должен) быть опущен. Вместо этого используется значение макроса `%buildroot`, который определён как `«%{_tmppath}/%{name}-buildroot»` в файле `/usr/lib/rpm/macros` и может быть переопределён в любом месте, где допускается определять макросы. В случае, если макрос `%buildroot` не определён либо его значение представляет собой недопустимое значение `«/»`, сборка пакета не будет выполнена.

Новые макросы

Встроенные макросы

`%homedir`

домашний каталог пользователя, вызывающего этот макрос;

`%homedir{user}`

домашний каталог пользователя `user`;

Макросы для часто используемых каталогов

manpages: `%_man1dir`, `%_man2dir`, `%_man3dir`, `%_man4dir`,
`%_man5dir`, `%_man6dir`, `%_man7dir`, `%_man8dir`, `%_man9dir`;

X11R6: `%_x11dir`, `%_x11bindir`, `%_x11libdir`, `%_x11includedir`,
`%_x11mandir`, `%_x11datadir`, `%_x11fontsdire`;

лицензии: `%_licensedir`;

меню: `%_menudir`, `%_iconsdir`, `%_miconsdir`, `%_liconsdir`;

emacs: `%_emacsispdir`;

tcl: `%_tcllibdir`, `%_tcldatadir`;

другие системные: `%_initdir`, `%_lockdir`, `%_logdir`, `%_cachedir`,
`%_spooldir`.

Управление опциями компилятора gcc

%add_optflags <options>

добавить указанные параметры в стандартный набор %optflags;

%remove_optflags <options>

убрать указанные параметры из стандартного набора %optflags;

%optflags_core

базовые параметры;

%_optlevel

уровень оптимизации;

%optflags_optimization

параметры, отвечающие за оптимизацию, кроме архитектурно-зависимых;

%optflags_warnings

warning options;

%optflags_debug

debugging options;

%optflags_shared

параметры, применяемые для создания relocatable файлов;

%optflags_nocpp

параметры, отключающие поддержку C++ exceptions и C++ RTTI;

%optflags_notraceback

-fomit-frame-pointer;

%optflags_fastmath

-ffast-math;

%optflags_strict

-fstrict-aliasing;

%optflags_kernel

параметры, используемые при компиляции ядра и его модулей.

По умолчанию, стандартный набор `%optflags` состоит из «`%optflags_core %optflags_warnings %optflags_optimization`».

Макросы-надстройки над утилитой make

%__nprocs

число процессоров, доступных для сборки с помощью `%make_build`;

%make_build

вызов `make` с параметром, обеспечивающим оптимальную параллельную сборку в данной среде;

%make_install

вызов `make` с инициализацией переменной `INSTALL`, что в случае корректной реализации `Makefile`-ов пакета позволяет сохранить дату последней модификации файлов, что особенно важно для документации;

%makeinstall

«`%make_install <инициализация других переменных, используемых многими Makefile-ами> install`»

Регистрация разделяемых библиотек

%post_ldconfig, %post_ldconfig_lib

регистрация новых/обновлённых библиотек;

%post_ldconfig_sys

регистрация новых/обновлённых системных библиотек (которые могут быть использованы в `chroot'ax`);

%postun_ldconfig

отмена регистрации удалённых библиотек.

Регистрация документации в формате info

%install_info

регистрация новых/обновлённых `info`-страниц;

%uninstall_info

отмена регистрации удалённых info-страниц.

Регистрация меню**%update_menus**

регистрация новых/обновлённых меню;

%clean_menus

отмена регистрации удалённых меню.

Регистрация каталогов scrollkeeper**%update_scrollkeeper**

регистрация новых/обновлённых каталогов;

%clean_scrollkeeper

отмена регистрации удалённых каталогов.

Вспомогательные макросы %configure**%__libtoolize**

путь к скрипту **libtoolize**;

%_configure_script

путь к скрипту **configure**;

%_configure_target

целевая платформа для **configure**;

%_configure_gettext

--without-included-gettext.

Серверные макросы**%post_service**

регистрация нового сервиса при установке, перезапуск при обновлении;

%preun_service

отмена регистрации сервиса и его выключение при удалении.

Макросы, определяющие некоторые аспекты packaging policy**%buildroot**

значение BuildRoot;

%_defattr

атрибуты файлов и каталогов по умолчанию для каждой секции %files и для каждого файла, включаемого в таких секциях;

%_cleanup_method

метод, используемый при удалении ненужных файлов в секции %install;

%_compress_method

метод, используемый при сжатии документации в секции %install;

%_findprov_default_method

метод, используемый по умолчанию при поиске предоставляемых зависимостей;

%_findreq_default_method

метод, используемый по умолчанию при поиске требуемых зависимостей;

%_fixup_method

метод, используемый при исправлении файлов в секции %install;

%_verify_elf_method

метод, используемый при проверке ELF-файлов в секции %install;

%_strip_method

метод, используемый при обработке ELF-файлов в секции %install;

%_ {cleanup,compress,fixup,strip,verify_elf,findreq,findprov}_topdir

точка начала поиска файлов, обрабатываемых соответствующим методом;

%_ {cleanup,compress,fixup,strip,verify_elf,findreq,findprov}_skiplist

список шаблонов файлов, которые будут пропущены при обработке соответствующим методом;

%set_ {cleanup,compress,fixup,strip,verify_elf}_method

изменить значение соответствующего макроса;

%set_ {cleanup,compress,fixup,strip,verify_elf,findreq,findprov}_ {topdir}

изменить значение соответствующего макроса;

%add_ {cleanup,compress,fixup,strip,verify_elf,findreq,findprov}_skiplist

добавить значение в соответствующий список.

Вызов вспомогательных программ

%find_lang

ВЫЗОВ **/usr/lib/rpm/find-lang**

%strip_executable

ВЫЗОВ **/usr/lib/rpm/brp-strip** для обработки *ELF executables*;

%strip_relocatable

ВЫЗОВ **/usr/lib/rpm/brp-strip** для обработки *ELF relocatables*;

%strip_shared

ВЫЗОВ **/usr/lib/rpm/brp-strip** для обработки *ELF shared objects*;

%strip_static

ВЫЗОВ **/usr/lib/rpm/brp-strip** для обработки *ELF ar archives*;

%cleanup_build

ВЫЗОВ **/usr/lib/rpm/brp-cleanup**;

%compress_docs

ВЫЗОВ **/usr/lib/rpm/brp-compress**;

%strip_binaries

вызов `/usr/lib/rpm/brp-strip`;

%clean_buildroot

выполнение `rm -rf %buildroot`, если `%buildroot` не указывает на настоящий `/`.

Управление процессом сборки**%buildmulti**

альтернативная директива `%build` для случая, когда в секции `%build` происходит заполнение `%buildroot`. Вообще говоря, такой техники стоит избегать во всех случаях, когда это возможно;

%_build_lang

значение переменных `LANG`, `LANGUAGE` и `LC_ALL`;

%_build_display

значение переменной `DISPLAY`;

%_build_xauthority

значение переменной `XAUTHORITY`;

%__ccache_cc

значение переменной `CCACHE_CC`;

%__ccache_dir

значение переменной `CCACHE_DIR`;

Версии некоторых установленных в системе пакетов

glibc: `%__glibc_version`, `%__glibc_version_major`,
`%__glibc_version_minor`;

gcc: `%__gcc_version`, `%__gcc_version_major`,
`%__gcc_version_minor`, `%__gcc_version_base`

python: `%__python_version`

%get_version

версия указанного пакета;

%get_release

релиз указанного пакета;

%get_serial

serial указанного пакета;

%add_serial

serial указанного пакета в виде, пригодном для включения в спес-файл;

%get_SVR

тройка значений `serial:version-release` указанного пакета;

%get_NSVR

четвёрка значений `name-serial:version-release` указанного пакета;

%get_dep

строка вида `name >= serial:version-release`, построенная по указанному пакету;

Эти макросы, как правило, используются в пакетах, сборка которых возможна с различными версиями этих программ, если эти версии правильно учитывать.

Управление процессом обработки спес-файлов

%def_with, %def_without, %def_enable, %def_disable

установка значения макросов условия с указанием значения по умолчанию;

%check_def

проверка макросов условия на непротиворечивость;

%subst_with, %subst_enable

подстановка значения макросов условия;

%defined, %undefined

проверка на существование макроса;

%with, %without, %enabled, %disabled

проверка значения макросов условия;

%ifdef, %ifndef

ветвление по факту существования макроса;

%if_with, %if_without, %if_enabled, %if_disabled

ветвление по значению макросов условия;

Прочие макросы**%intel**

список архитектур Intel™, совместимых с i386;

%amd

список архитектур AMD™, совместимых с i386;

%ix86

список всех архитектур, совместимых с i386;

компоненты макроса %packager

%packagerName, %packagerAddress

%_internal_gpg_path

путь к связке ключей *ALT Linux Team*.

Новые параметры rpm**-bE**

новый режим работы RPM, при котором происходит только подстановка макросов;

--nowait-lock

не блокировать процесс, если база данных RPM занята;

--fancypercent

отображать дополнительную информацию о процентах проделанной работы при установке/обновлении пакетов;

--nopatch

не включать указанные патчи в исходный пакет;

`--nosource`

не включать указанные исходники в исходный пакет;

`--lastchange`

вывести информацию о последнем изменении пакета;

`--changes-since`

вывести информацию обо всех изменениях пакета, начиная с указанной версии.

Новые возможности rpm по сборке пакетов

По окончании выполнения секции `%install` RPM выполняет ряд действий:

- удаление ненужных файлов и каталогов;
- исправление прав доступа к файлам и каталогам;
- упаковка документации;
- удаление отладочной информации;
- коррекция символических ссылок на разделяемые библиотеки;
- перекомпиляция python-модулей.

Автоматическое удаление ненужных файлов

Все файлы и каталоги, подпадающие под правило определения ненужных файлов и каталогов, удаляются. В частности, по умолчанию подлежат удалению

- файлы с именами `DEADJOE`, `.SUMS`, `TAGS`, `core`;
- файлы, заканчивающиеся на `~`, `.orig`, `.rej`, `.bak`;
- каталоги с именем `CVS`.

Поддерживаются следующие методы определения файлов и каталогов, подлежащих удалению:

`none, skip`

поиска и удаления не производится;

`auto`

метод по умолчанию, определенный в файле `/usr/lib/rpm/brp-cleanup`;

*

специальный метод; переданное значение используется в качестве имени программы, которая будет вызвана для поиска и удаления ненужных файлов.

Какой метод будет использован в каждом конкретном случае, зависит от значения макроса `%_cleanup_method`; значение по умолчанию для этого макроса — `auto`.

Автоматический поиск и исправление конфигурационных файлов, используемых прежде всего при разработке ПО

Поддерживаются следующие типы файлов, подлежащих проверке и исправлению:

`none, skip`

поиска и проверки не производится;

`binconfig`

поиск и обработка shell-скриптов по шаблону `/usr/bin/*-config`;

`pkgconfig`

поиск и обработка файлов по шаблону `/usr/lib/pkgconfig/*.pc`;

`libtool`

поиск и обработка `.la`-файлов;

Какой метод будет использован в каждом конкретном случае, зависит от значения макроса `%_fixup_method`; значение по умолчанию для этого макроса — `binconfig pkgconfig libtool`.

Автоматическое исправление прав доступа к файлам и каталогам

Права доступа ко всем файловым объектам, находящимся в `$RPM_BUILD_ROOT`, проверяются и корректируются согласно следующим правилам:

- каталоги `/usr/share`, `/usr/include`, `/usr/X11R6/share`, `/usr/X11R6/include`, `/usr/X11R6/man` со всем содержимым должны быть доступны по чтению всем пользователям;
- ничто из содержимого каталога `/usr`, за исключением `/usr/src`, не должно быть доступно по записи не-владельцу, за исключением владельца файлов.
- никакие `suid` и/или `sgid`-файлы не должны быть доступны по чтению (и тем более по записи), за исключением владельца файлов.

Автоматическое сжатие man- и info-документации с поддержкой различных методов сжатия

Вся документация пакета, распознаваемая как man- или info-документация, по окончании работы секции `%install`, сжимается согласно выбранному методу. Поддерживаются следующие методы сжатия:

`bzip2`

сжатие с помощью `bzip2 -9`;

`gzip`

сжатие с помощью `gzip -9n`;

`auto`

сжатие с помощью `gzip -9n` либо `bzip2 -9` в зависимости от того, какой вариант окажется эффективнее;

`none`

производится декомпрессия файлов вместо сжатия;

`skip`

процедура сжатия пропускается полностью.

Какой метод будет использован в каждом конкретном случае, зависит от значения макроса `%_compress_method`; значение по умолчанию

для этого макроса — `auto`. По окончании процедуры сжатия производится выравнивание ссылок, которые, возможно, требуют коррекции в связи с изменениями имён файлов в процессе их сжатия.

Автоматическая проверка ELF-файлов с поддержкой различных стратегий

Иногда в результате сборки пакета получаются ELF-файлы, содержащие неверную и/или недопустимую информацию в некоторых секциях, таких как `RPATH`. Поэтому по окончании работы секции `%install` проверяются все собранные ELF-файлы. Выбор типов файлов определяется значением макроса `%_verify_elf_method`, которое есть набор из следующих возможных значений:

`none, skip`

поиска и проверки не производится;

`relaxed`

проверка только на наличие недопустимых элементов в `RPATH`;

`normal`

`relaxed` + проверка на наличие более чем одного элемента в `RPATH`;

`strict`

проверка на наличие непустого `RPATH`.

Значение по умолчанию для макроса `%_verify_elf_method` в данный момент равно `normal`.

Автоматическое удаление отладочной информации из ELF-файлов с поддержкой различных стратегий выбора файлов, подлежащих обработке

Зачастую возможно уменьшить размер получаемых в результате сборки пакета ELF-файлов без потери качества за счёт удаления из них отладочной информации. Поэтому по окончании работы секции `%install` все ELF-файлы выбранных типов обрабатываются программой `strip`. Выбор типов файлов определяется значением макроса `%_strip_method`, которое есть набор из следующих возможных значений:

`executable`

ELF executable;

`relocatable`

ELF relocatable;

`shared`

ELF shared object;

`static`

ar archive.

Кроме того, есть возможность вызывать `strip` вручную, для этой цели предназначены макросы `%strip_executable`, `%strip_relocatable`, `%strip_shared`, `%strip_static`. Синтаксис этих макросов подробно изложен в `/usr/lib/rpm/brp-strip --help`.

Автоматическая перекомпиляция python-модулей

Как известно, python-модули обычно компилируют в байтовую форму для увеличения быстродействия при последующей работе с ними. Каждый такой модуль, помимо всего прочего, хранит время своего создания и полное имя файла, в котором должен находиться. В связи с последним обстоятельством скомпилированные модули, созданные в результате работы секции `%install`, непригодны, ибо не могут быть использованы после установки пакета. По этой причине теперь по окончании работы секции `%install` производится перекомпиляция всех python-модулей таким образом, чтобы их можно было использовать после установки пакета. В качестве байт-компилятора будет использоваться программа, имя которой хранится в макросе `%__python`. Обычно это `/usr/bin/python`, однако в некоторых случаях может потребоваться изменить это значение на другое (например, в случае сборки пакета `python` или если по какой-то причине перекомпиляция не нужна).

Автоматический поиск требуемых и предоставляемых зависимостей

В дополнение к стандартному поиску зависимостей от/для разделяемых библиотек, реализована поддержка поиска требуемых зависимостей для shell- и perl-скриптов, поиска зависимостей, определяемых наличием специальных файлов в пакете, а также поддержка поиска предоставляемых зависимостей для perl-скриптов.

Изменение семантики тэгов, управляющих поиском зависимостей

Новые возможности RPM по автоматическому поиску зависимостей при сборке пакетов управляются, как и прежде, значениями тэгов `AutoReq`, `AutoProv` и `AutoReqProv`. К стандартным значениям `yes/no` (`true/false`), таким образом, добавлены новые возможные значения, являющиеся именами методов поиска зависимостей:

`lib/nolib`

включение/выключение поиска зависимостей от/для разделяемых библиотек;

`shell/noshell`

включение/выключение поиска зависимостей в shell-скриптах;

`perl/noperl`

включение/выключение поиска зависимостей в perl-скриптах;

`files/nofiles`

включение/выключение поиска зависимостей, определяемых наличием специальных файлов в пакете;

`default`

то же, что и `yes`;

`none, off`

то же, что и `no`;

`all`

включение всех возможных методов поиска зависимостей.

Значением тэга может являться как один метод, так и перечисление методов. По умолчанию, для каждого подпакета собираемого пакета `AutoReq = AutoProv = yes`, что на практике означает использование макросов `%_findreq_default_method` и `%_findprov_default_method` для определения методов поиска зависимостей.

Автоматическая очистка BuildRoot

Перед выполнением секции `%install` и по окончании выполнения секции `%clean` RPM автоматически очищает `BuildRoot` с помощью макроса `%clean_buildroot`. Это значит, что больше не нужно использовать

эти ужасные `rm -rf $RPM_BUILD_ROOT`. Секция `%clean` вообще может (и должна) быть опущена, если в ней не содержится ничего, кроме этого «`rm`». В тех редких случаях, когда в спес-файле производится заполнение `BuildRoot` не в секции `%install`, как это должно быть, а в секции `%build`, что в принципе неправильно, можно перенести точку очистки `BuildRoot` из начала секции `%install` в начало секции `%build`, если заменить директиву `%build` на макрос `%buildmulti`.

Упрощение секции `%files`

Ранее в начале каждой секции `%files` было необходимо указывать атрибуты файлов и каталогов создаваемых пакетов с помощью довольно однообразно используемой директивы `%defattr`. Теперь это происходит автоматически в начале каждой секции `%files`, а также в начале каждого файла, включаемого в секцию `%files` с помощью опции `-f`. Точнее говоря, в качестве этой директивы используется значение макроса `%_defattr`. Таким образом, прежнее использование директивы `%defattr` в начале секций и файлов следует считать упрзднённым.

Сборка пакетов привилегированным пользователем

То, что когда-то было необходимостью, со временем стало излишним, а порой и просто опасным. Теперь, когда все без исключения пакеты можно (и нужно) собирать непривилегированным пользователем во избежание риска разрушения системы и некорректной сборки, сборка пакетов привилегированным пользователем по умолчанию запрещена. Этот запрет можно снять путём изменения значения макроса `%_allow_root_build`.

Литература

[wwwrpm] Официальный web-сайт rpm: <http://www.rpm.org/> .

[mailrpm] Список рассылки для разработчиков rpm: `rpm-list@redhat.com` .

[maxrpm] Edward Bailey. February 17, 1997. Maximum RPM, (доступна также online-версия по адресу <http://www.rpm.org/max-rpm/> и в формате PostScript по адресу <http://www.rpm.org/local/maximum-rpm.ps.gz>).

ALT Secure Packaging Policy

Дмитрий Левин

Преобразование оригинального текста в *DocBook* : Юрий Зотов

<yznews@hotmail.ru>

Массовые операции над файлами и каталогами (секции: %setup, %build, %install, %pre*, %post*, %trigger*)

При массовой обработке файлов и каталогов (glob expansion, find и др.) *НЕОБХОДИМО* отделять команду с параметрами от списка аргументов разделителем «--» везде, где это поддерживается.

Обоснование: Массовые операции над файлами, имена которых начинаются на «-», могут давать неверный результат в случае неиспользования «--».

При использовании утилиты **find** для изменения файлов и каталогов *НЕОБХОДИМО* использовать параметр `-print0`; соответствующие ему параметры других утилит:

xargs: `-r0`

grep: `-Z`

sort: `-z`

Обоснование: Использование **find** при работе с каталогами, содержащими объекты с нестандартными именами (пробелами и др.), без использования `-print0` приводит к неправильному результату.

Пример 1.1. Правильное использование **find**

```
find -type f -print0 |
  xargs -r0 %__grep -FZl 'mawk gawk' -- |
  xargs -r0 %__perl -pi -e 's/mawk gawk/gawk mawk/g' --
```

Операции с временными файлами

При необходимости создания временных файлов и/или каталогов следует использовать утилиту **mktemp(1)** совместно с командой **trap**, например:

```
TMPTFILE="mktemp -t somename.XXXXXXXXXX" || exit 1
exit_handler()
{
  local rc=$?
  trap '' EXIT
  rm -f -- "$TMPTFILE"
  exit $rc
}
```

```
}
trap exit_handler EXIT HUP INT PIPE TERM QUIT
```

Не следует пользоваться фиксированными либо предсказуемыми именами для создания временных файлов в общедоступных каталогах, таких как `/tmp`. Не следует оставлять временные файлы в случае успешного окончания текущей стадии сборки пакета.

Чужие и системные каталоги и файлы (секции: `%install`, `%files`)

Пакеты *НЕ ДОЛЖНЫ* включать в свой состав чужие каталоги и файлы, в частности, системные объекты файловой системы, а также файлы устройств (последнее — прерогатива пакета `dev`).

Обоснование: У каждого объекта файловой системы, имеющего отношение к дистрибутиву, должен и может быть только один владелец (или группа родственных владельцев в случае, когда несколько подпакетов одного пакета совместно используют общий каталог). Это лучше обеспечивает управление атрибутами объектов файловой системы, а также решает многие проблемы определения сборочных зависимостей между пакетами.

Атрибуты файлов и каталогов (секции: `%install`, `%files`)

Права доступа на привилегированные исполняемые файлы

Привилегированные исполняемые файлы, т.е. исполняемые файлы с установленными битами `suid` и/или `sgid`, *НЕ ДОЛЖНЫ* быть доступны по чтению (и тем более по записи) кому-либо, кроме процессов с `uid=0`.

Разделы, предназначенные для использования `readonly`

Пакеты *НЕ ДОЛЖНЫ* содержать файлы и каталоги в поддереве `/usr`, разрешающие доступ по записи кому-либо, кроме процессов с `uid=0`. Более того, программам, входящим в пакет, во время своей работы *НЕ СЛЕДУЕТ* полагаться на то, что файловые объекты, находящиеся в `/usr`, доступны по записи.

Файлы и каталоги, доступные для записи

Пакеты *НЕ ДОЛЖНЫ* содержать файлы и каталоги, доступные для записи всем пользователям. Должны быть предусмотрены меры по разграничению доступа, например, путём предоставления доступа по записи определённой группе(ам).

Владельцы файлов

Пакеты *НЕ ДОЛЖНЫ* содержать файлы, принадлежащие псевдо-пользователям, если в процессе работы к этим файлам осуществляется доступ процессов с другим `uid` либо с более широкими правами доступа. К таким файлам, в частности, относятся исполняемые, конфигурационные и неизменяемые файлы.

Обоснование: Псевдо-пользователь не должен иметь право изменять эти файлы; нарушение этого правила, как правило приводит к очевидной возможности осуществления `pseudouser/root compromise`.

Владельцы каталогов

Пакеты *НЕ ДОЛЖНЫ* содержать каталоги, принадлежащие псевдо-пользователям. Вместо этого следует использовать каталоги, принадлежащие `root`, с установленным `sticky bit` и доступом группы по записи.

Обоснование: Псевдо-пользователь не должен иметь право изменять атрибуты каталогов, а также файлы и каталоги, созданные другими пользователями; нарушение этого правила, как правило приводит к возможности осуществления `pseudouser/root compromise`.

Блокировки (секции: `%build`, `%install`, `%files`)

Разные пакеты, использующие блокировки для работы с общими файловыми объектами, такими как `tbody`'ы, во избежание потери данных *ДОЛЖНЫ* придерживаться единого механизма блокировки.

Например, для блокировки `tbody`'ов *НЕОБХОДИМО* использовать метод, за которым закрепилось имя `fcntl`. Не допускается использование привилегированных программ для `dotlocking`'а.

Обоснование: Каждая привилегированная программа — это дополнительная степень риска для системы, в которой такая программа установлена. Поэтому следует минимизировать потребность в подобных средствах. Метод блокировки `fcntl` опирается на системный вызов `fcntl(2)`, удовлетворяющий стандарту POSIX, и, следовательно, более широко распространённый, чем его аналог `flock(2)`.

Глава 2. ALT Linux Maintainer HOWTO

Перевод в формат *DocBook* : Юрий Зотов

<yznews@hotmailbox.ru>

Об этом документе

Этот документ предназначен в помощь тем, кто решил присоединиться к команде *ALT Linux Team*. Здесь вы не найдете полной информации о том, что такое проект *ALT*, что такое *Sisyphus*², как следует правильно собирать пакеты — это удел соответствующих *HOWTO*.

Команда разработчиков *ALT Linux Team*

Вот несколько основных положений об *ALT Linux Team*:

- *ALT Linux Team* (в дальнейшем просто *ALT*) — международная команда разработчиков;
- *ALT* не занимается противозаконной деятельностью. Это не политическая организация, а сообщество творческих людей;
- *ALT* занимается любыми проектами, не противоречащими ее идеологии. На данный момент предпочтение отдается программным проектам; Open Source, однако, если у кого-либо возникнет какая-нибудь новая, интересная и не противоречащая уголовному кодексу идея, то милости просим;
- Участие в *ALT* не накладывает на участников никаких ограничений и обязательств, разве что кроме диктуемых собственной совестью каждого. Можно параллельно участвовать в других проектах/организациях/партиях;
- Присоединиться или покинуть *ALT* может любой человек по собственному желанию и в любой момент времени. Когда будут открыты новые заселенные разумом планеты, то инопланетяне также смогут участвовать в этом проекте;

Среди участников особо выделяются:

²<http://www.altlinux.ru/index.php?module=sisyphus>

Принимающие

занимаются процедурами регистрации и отмены оной участников *ALT*;

Security Officer

занимаются вопросами безопасности основного репозитория. Следят за ошибками, связанными с безопасностью в пакетах и оповещают о них всех пользователей и участников *ALT*;

Sisyphus gate-keeper

стоят перед входом в *Sisyphus*, так называемым *Incoming*, и проверяют пакеты на соответствие правилам, принятым внутри *ALT*;

Более подробная информация об *ALT Linux Team* находится в соответствующем *HOWTO*.

Sisyphus — основной репозиторий пакетов *ALT Linux Team*

Sisyphus — это основной репозиторий пакетов *ALT Linux Team*³. Все наработки участников *ALT* хранятся здесь.

Минимальная единица хранения — пакет. Пакет может быть бинарным (содержать исполняемые модули) или с исходным кодом.

Формат пакета один для всего репозитория

- На данный момент основным форматом является RPM;
- Бинарные пакеты собираются под архитектуру i586;
- Поле `Vendor` должно содержать "ALT Linux Team";
- Обязательно должен быть `Changelog` с заголовком в формате

`<дата> <мантейнер> <его e-mail> <номер версии и сборки (release)>`

- Обозначение сборки должно быть в формате `alt<номер>` для новых пакетов и `ipl<номер>mdk` для старых, переименование со стандарта `ipl*mdk` на `alt*` может происходить только при увеличении версии пакета или при переписывании программы заново;

³<http://www.altlinux.ru>

Для каждого пакета определен один или несколько мантейнеров — участники *ALT*, которые следят за его актуальностью и работоспособностью (об обязанностях мантейнеров см. соответствующий раздел данного HOWTO). В каждом пакете есть поле, указывающее на его мантейнера.

Замечание

На данный момент поле `Packager` в формате RPM указывает либо на мантейнера (в случае, когда он один), либо на последнего собиравшего пакет (в случае групповой разработки).

Sisyphus может подразделяться на несколько репозиториев. На данный момент существуют:

- Основной репозиторий, собственно *Sisyphus*;
- репозиторий `contrib` — сюда попадают все новые и не проверенные пакеты;
- репозиторий `classic` — объединяет все пакеты, имеющиеся как в основном репозитории, так и в `contrib`. Создан для обратной совместимости со старой схемой *Sisyphus* (до 1 июня 2002);
- Собрание устаревших пакетов (`obsoletes`) — сюда попадают пакеты с исходным кодом программ, которые морально устарели и в силу этого уже не пересобираются в современном окружении, или которые вытеснены другими (более современными пакетами). Последнее возможно только по личному желанию мантейнера (об этом чуть позже);
- Собрание неподдерживаемых пакетов (`unsupported`) — пакеты, которые в силу каких-либо причин (чаще всего это лицензионные ограничения) *ALT Linux Team* не может поддерживать в полном объеме. Пакеты могут присутствовать как в виде бинарных, так и в виде пакетов с исходным кодом. В отдельных случаях исходный код может отсутствовать в пакете — это готовая заготовка, из которой вы можете собрать бинарный пакет;
- Собрание брошенных пакетов (`orphaned`) — пакеты с исходными текстами, которые на данный момент не поддерживаются. Если вы не знаете, за какой пакет взяться, то посмотрите прежде всего в этот каталог;

ALT Linux Team не гарантирует работоспособности входящих в *Sisyphus* пакетов. Это полет мысли, текущая разработка, а не готовый к употреблению дистрибутив.

Если мантейнер совершенно не уверен в качестве своей программы и опасается класть ее в *Sisyphus*, то существует отдельный репозиторий

для таких «экстремальных» пакетов — *Daedalus*. На данный момент туда зачастую попадают нестабильные сборки и *alpha*-версии пакетов.

Для каждого пакета существует четко определенная схема попадания в репозиторий. Исходной точкой является *incoming*, конечной — репозиторий *Sisyphus*. Маршрут определяется используемой на данный момент технологией.

Перед тем, как попасть в *Sisyphus*⁴, пакет обязательно проходит ручную проверку специально выделенными для этого участниками *ALT* (*incominger*). Пакет проверяется на качество и соответствие правилам сборки и ему может быть отказано в доступе в репозиторий. Если пакет не прошел какой-либо из участков маршрута в репозиторий, то мантейнеру посылается уведомление об этом с указанием причины.

Для каждого репозитория *Sisyphus* может существовать отдельный *incominger*.

Пакет может поменять своего мантейнера по одной из следующих схем:

- Если он находится в *orphaned* более одной недели, то пакет можно взять без чьего-либо разрешения. Перед тем, как положить этот пакет в *incoming*, его необходимо пересобрать с увеличением номера версии или релиза;
- Если пакет не находится в *orphaned*, то пакет можно взять, получив разрешение у текущего (последнего) мантейнера или группы оных (определяется по полю *Packager*);
- Если мантейнер не откликается после 5 запросов, то пакет автоматически переходит в *orphaned*;

Все репозитории *Sisyphus* имеют разный уровень надежности. Далее перечислены текущие репозитории в порядке убывания надежности:

- Основной
- *contrib*
- *orphaned*
- *unsupported*
- *obsoletes*

Перемещение пакета из менее надежного репозитория в более надежный может происходить только после ручной проверки проверяющим (*incominger*) более надежного репозитория.

Перемещение из более надежного репозитория в менее надежный может происходить автоматически.

⁴<http://www.altlinux.ru/index.php?module=sisyphus>

Прием новых участников в *ALT Linux Team*

Прием производят специально выделенные участники *ALT* — *принимающие*.

Претендент посылает к одному из этих *принимающих* запрос на включение пакета в *Sisyphus* или *Daedalus*, с указанием причины, побудившей его сделать это.

Принимающий подтверждает получение уведомления, возможно, после совещания с остальными участниками.

Претендент высылает *принимающему* первичную информацию о себе. Информация посылается по *e-mail* на адрес <join@altlinux.ru> На данный момент это:

- Псевдоним (имя пользователя) участника, выбирается им самим, если еще не принадлежит существующему участнику. Его длина должна быть по возможности минимальной и он не должен содержать цифры;
- Создается новый *OpenSSH* ключ (DSA, 2048 бит). *Принимающему* высылается публичная часть ключа. Этот ключ будет необходим для выкладывания пакетов в *incoming* (точка входа в репозиторий *Sisyphus*);

Замечание

Ключ настоятельно рекомендуется сделать с паролем. В противном случае нет никакой гарантии в том, что никто не испортит ваш пакет, действуя от вашего имени.

- Создается/модифицируется существующий GPG/PGP ключ (DSA and ElGamal, 2048 бит). В ключе должен быть *uid* вида <псевдоним>@altlinux.ru. *Принимающему* высылается его публичная часть

Ключ высылается в ASCII виде. Экспортирование выполняется следующей командой:

```
gpg --export --armor [имя ключа] >[файл с экспортированным в_
ASCII ключом]
```

- Указывается адрес, на который будет пересылаться почта с адреса <псевдоним>@altlinux.ru

Принимающий на основе переданной первичной информации регистрирует нового участника и высылает ему инструкции по использованию `incoming`, тестовое задание, возможно, дополнительную информацию.

Инструкция по использованию `incoming` необходима, так как `incoming` — это первичная точка входа в репозиторий для всех пакетов. Инструкция может видоизменяться в зависимости от используемой на данный момент технологии.

Тестовое задание носит обучающий характер, позволяет новому участнику освоиться с основными принципами работы с `incoming` и создания пакетов. Тестовое задание зависит от того, какой пакет хочет в дальнейшем обслуживать новый участник. Никакие пакеты нового участника не будут приниматься до тех пор, пока он не выполнит тестовое задание.

После успешного прохождения тестового задания *принимающий* (не обязательно именно тот, который проводил регистрацию) производит при необходимости дополнительную регистрацию, на данный момент это подписывание на список рассылки `<devel@altlinux.ru>`.

На первый период *принимающий* назначает одного из существующих мантейнеров в помощь новому участнику.

Обязанности мантейнера

Что должен делать мантейнер:

- Следить за современностью и актуальностью поддерживаемых им пакетов. Регулярно пользоваться этим пакетом (мантейнер должен чувствовать пакет «изнутри», без этого нарушается цикл тестирования и пакет становится неполноценным);
- Быть в курсе разработки софта, входящего в пакет. Это, как минимум, подразумевает участие в списке рассылки типа *XXX-announce*. Общение с разработчиками софта, входящего в пакет, желательно, но не обязательно. Участие в разработке софта, входящего в пакет, желательно, но не обязательно. (это дает возможность иметь в дистрибутиве самую свежую, но при этом рабочую версию софта, а также повышает оперативность исправления ошибок);
- Незамедлительно исправлять ошибки, связанные с безопасностью по первому запросу *Security Officer*;
- По мере возможности исправлять ошибки, связанные с некорректным функционированием программ;
- Помогать прикрепленным к нему новым участникам;

- По возможности активно участвовать в списках рассылок *ALT*;

Работа с ключами

При приеме участник создает два ключа (на данный момент это *OpenSSH* и GPG ключи).

При утере одного из ключей, он может быть восстановлен заверением вторым. Берегите свои приватные ключи!

При утере обоих ключей участник извещает об этом *принимающих*. Его доступ в *incoming* прекращается до восстановления ключей.

Два ключа могут быть восстановлены либо посредством личной встречи с одним из *принимающих*, либо посылкой их письмом, заверенным ключом одного из участников *ALT Linux Team*. В последнем случае всю ответственность за дальнейшую безопасность репозитория несет участник, заверивший ключи.

Устройство *incoming*

Возможно существование иерархии *incoming* для *Sisyphus*, но основной каталог только один. Структура его четко определена и мантейнеры обязаны класть пакеты в определенные каталоги, предназначенные для этого.

Достаточно класть только пакеты с исходными текстами, так как они всегда проходят пересборку перед тем, как попасть в основной репозиторий.

Итак, *incoming* подразделяется на следующие составляющие:

- **Sisyphus** — здесь должны появляться пакеты с исходными текстами, предназначенные для репозитория. Бинарные пакеты кладутся в подкаталог *bin*;
- **Daedalus** — здесь должны появляться пакеты с исходными текстами и бинарные пакеты, предназначенные для репозитория *Daedalus*;
- **FROZEN** — когда начинается разработка дистрибутива, для последнего создается отдельное дерево с исходными и бинарными пакетами. Если мантейнер хочет, чтобы его пакет попал в это дерево, но не в *Sisyphus*, то класть пакет нужно именно в этот каталог;

Пожалуйста, придерживайтесь этой структуры каталогов, не создавайте своих подкаталогов. *Sisyphus gate-keeper* оставляет за собой право не брать пакет, если он лежит в неправильном месте.

В помощь начинающему разработчику

Возможные причины отказа в доступе к *Sisyphus* для пакета

Вот несколько причин, по которым пакету может быть отказано в доступе в репозиторий (более подробные сведения находятся в ALT-Packaging-HOWTO):

- Несоответствие требованиям репозитория (см. выше);
- Пакет не подписан GPG-ключом мантейнера;
- В пакете недопустимые `%post`, `%preun`, `%pre` скрипты. Например:
 - нельзя изменять на этапе установки/удаления какие-либо системные файлы;
 - не допускается доустанавливать дополнительные программы, производить их перемещения из одного места в другое;
- Недопустимые права на файлы (немотивированный SUID/SGID, *World-writable* файлы);
- Ложные или недопустимые зависимости пакета (Requires);
- Ложные или недопустимые зависимости сборки (BuildReqs);
- Полное несоответствие *spec*-файла требованиям *ALT Linux Team* (более подробно в ALT-Packaging-HOWTO). Например, использование `BuildRoot`, `%clean`, `defattr(-, -, root)`;
- Избыточная информация в версии пакета (например, `1.2.3pre5`) может повредить корректному обновлению (`1.2.3` до `1.2.3pre5`, несмотря на то, что `1.2.3` — это финальная версия). Переносите все дополнительные сведения в номер сборки (например, `alt0.1.pre5`);
- Необоснованное увеличение *Serial*. Увеличение следует производить только в случае необходимости отката по версии;

Примерное содержание файла `.rpmmacros`

Очень удобно и настоятельно рекомендуется создать в своем домашнем каталоге файл `.rpmmacros`. В нем вы можете определить такие параметры, как:

- Месторасположение структуры каталогов для сборки;
- Каким ключом подписывать пакеты;
- Значение полей, таких, как `Packager`, `Vendor` и `Distribution`;

```
%vendor ALT Linux Team
%distribution ALT Linux

%_topdir /home/pupkin/RPM/
%_gpg_name pupkin@altlinux.ru
%packager Vasja Pupkin <pupkin@altlinux.ru>
```

Типовая инструкция, высылаемая новому мантейнеру

Greetings!

Ваш account в incoming Sisyphus создан (название account'a in_xxx). О том как им пользоваться, смотрите инструкции ниже. Также создан forward с адреса xxx@altlinux.ru на yyy@somehost.com

Вашим главным помощником будет zzz@altlinux.ru (ZZZ ZZZZ) - если у Вас возникнут какие-либо вопросы по изготовлению пакетов спрашивайте у него.

ПРЕЖДЕ ЧЕМ ВЫ НАЧНЕТЕ ВЫКЛАДЫВАТЬ ПАКЕТЫ:

Надо пройти простенький тест. Это поможет Вам лучше освоиться с технологией работы с incoming, избежать типичных ошибок при сборке пакетов, с'экономить деньги на лишнем трафике. Итак, Задача:

Необходимо собрать пакет, удовлетворяющий следующим требованиям:

1. xxxxxxxxxxxxxx
2. xxxxxxxxxxxxxx
3. xxxxxxxxxxxxxx
4. xxxxxxxxxxxxxx
5. xxxxxxxxxxxxxx

После того как Вы успешно сделаете этот пакет, можете выкладывать то, что хотите.

Инструкции:

Теперь можно добавить в `~/.ssh/config`:

```
Host incoming
  HostName cvs.altlinux.org
  User in_xxx
  Protocol 2
  ForwardX11 no
  ForwardAgent no
  Compression no
```

`incoming` - это специальная среда с `rsync`'ом, и `ssh` используется прежде всего как транспорт для `rsync`.

В каталог `/incoming` заливаются новые пакеты, собранные разработчиками, находящимися вне офиса.

Для закачки туда-сюда файлов лучше всего использовать `rsync`.
Например,
для `upload`'а в `incoming` можно запускать нечто вроде

```
rsync -va --partial --stats -e ssh local_path/
incoming:/incoming/Sisyphus/
```

Если что-то непонятно либо не получается - спрашивайте.

Тест на работоспособность:

```
$ ssh incoming
-in_sh: invalid arguments
Connection to cvs.altlinux.org closed.
```

```
$ rsync -n -e ssh incoming:/incoming/
drwxr-xr-x      4096 2003/02/05 16:58:31 .
drwxr-sr-t      4096 2003/05/15 15:53:25 Daedalus
drwxr-s--T     16384 2003/05/19 13:20:04 Sisyphus
drwxr-s--T      4096 2003/05/13 04:56:39 ivk
drwxr-xr-x      4096 2003/01/23 18:13:10 release
drwxr-xr-x      4096 2003/03/28 16:45:21 updates
wrote 24 bytes  read 135 bytes 106.00 bytes/sec
total size is 0  speedup is 0.00
```

С наилучшими пожеланиями
GATE-KEEPER NAME

Глава 3. Принципы создания документов для *ALT Linux Team Documentation*⁵

Обоснование выбора *DocBook*

После очередного ответа на повторяющиеся вопросы стало, наконец, ясно, что информация есть — её нужно упорядочить и сделать доступной. Для упорядочивания и структурирования информации хорошо подходит формат *DocBook*. *XML*⁶ — это подмножество *SGML* и *Docbook*⁷ — тип документа (DTD), разработанный для написания документации. Существуют *DTD Docbook*⁸ для *XML* и *SGML*, но выбран именно *XML*, как более прогрессивный.

Этот формат для документации является аналогом исходных кодов для программ, однажды созданный документ можно преобразовывать в другие форматы, например, *html*, *pdf*, *rtf*, *tex*, *man*, *dvi*, *ps* и другие. Инструменты обработки *DocBook* автоматизируют создание индексов, оглавлений, перекрёстных ссылок, включение внешних документов, изменения сущностей средствами языков преобразований (*DSSSL* и *XSL*) и многое другое — более детальную информацию можно найти в разделе Информация о *DocBook*.

Другие предлагавшиеся форматы, например, *LaTeX*, являются ориентированными на определённое использование и поэтому не подходят для создания универсальной документации. *DocBook* используется для смысловой разметки текста, поэтому конвертация в него из других форматов недопустима.

Замечание

Документация должна быть корректно написана и структурирована, и *НЕ* должна ориентироваться на способ публикации, печатный или сетевой.

Существуют гибкие средства преобразования документов в формате *DocBook*, позволяющие из одного и того же документа получать результаты для разных целей. Например, можно помечать часть текста как общую, часть — как специфичную для *ALT Linux Team*, и

⁵<http://docs.altlinux.ru>

⁶<http://www.rol.ru/news/it/helpdesk/xml01.htm>

⁷<http://www.docbook.org>

⁸<http://www.docbook.org/tdg/en/html/docbook.html>

получать как универсальный, так и специфичный для *ALT* результат. Такой приём называется *Profiling*⁹. В *DocBook* стоит создавать только структурированные, продуманные документы, которые будут развиваться и использоваться длительное время, для простых текстов, например, электронных писем, он слишком трудоёмок.

Инструменты для создания документов

DocBook довольно многословен, поэтому писать в нём руками неудобно. Инструменты, облегчающие написание документов:

Emacs PSGML mode

Режим для Emacs, облегчающий редактирование XML и SGML документов. Поддерживает любые описания типов документов (DTD), выравнивание в зависимости от уровня вложенности, автоподстановку, автодополнение тегов и конструкций. Является одним из самых мощных средств для работы с XML документами, но сложен и требует начального обучения. Данный текст написан с использованием режима PSGML. В *Sisyphus* находится в пакете `emacs-mode-psgml`, подробная информация доступна на *домашней странице*¹⁰;

Emacs xslide mode

Режим для Emacs, облегчающий редактирование XSL стилей. Поддерживает выравнивание в зависимости от уровня вложенности, автоподстановку, автодополнение тегов и конструкций. В *Sisyphus* находится в пакете `emacs-mode-xslide`, подробная информация доступна на *домашней странице*¹¹;

KXML Editor

Редактор XML для KDE. Графически отображает структуру документа, но неудобен для редактирования легко читаемых текстов, подходит скорее для редактирования файлов конфигурации в формате XML. Не поддерживает DTD. В *Sisyphus*¹² находится в пакете `kxmleditor`, подробная информация доступна на *домашней странице*¹³;

⁹<http://docbook.sourceforge.net/release/xsl/current/doc/tools/profiling.html>

¹⁰http://www.lysator.liu.se/projects/about_psgml.html

¹¹<http://www.menteith.com/xslide/>

¹²<http://www.altlinux.ru/index.php?module=sisyphus>

¹³<http://kxmleditor.sourceforge.net/>

LyX

Очень удобное средство для работы с документами, к сожалению, пока сохраняет документы только в устаревшем формате *DocBook/SGML*, и нет достоверной информации, что может работать с *DocBook*. В *Sisyphus* находится в пакете `lyx`, подробная информация доступна на *домашней странице*¹⁴;

tkXMLive

Визуальный редактор XML документов, написан на языке tcl, находится в стадии разработки. Подробная информация доступна на *домашней странице*¹⁵;

Преобразование в другие форматы

Для преобразования исходных документов в другие форматы используются языки описания преобразований и таблицы стилей. Существуют два языка для работы с таблицами стилей (*stylesheets*):

DSSSL

Язык семантики и описания стиля документа (DSSSL — Document Style Semantics and Specification Language) был создан для работы с документами в формате SGML и может работать с документами в формате XML, поскольку второй является подмножеством первого. В *Sisyphus* *таблицы стилей DSSSL для работы с Docbook*¹⁶ находятся в пакете `docbook-style-dsssl`.

*Утилиты*¹⁷, использующие эти таблицы стилей, находятся в *Sisyphus* в пакете `docbook-utils`.

Пример 3.1. Преобразование XML-документа в формат HTML с использованием DSSSL-стилей

```
db2html file.xml
```

XSL

Расширяемый язык стилей (XSL — Extensible Style Language) — язык обработки стилей, созданный специально для работы с XML. Он позволяет преобразовывать документы из XML в HTML, RTF, а также в другие документы XML. Для преобразования XML-документов

¹⁴<http://www.lyx.org>

¹⁵<http://tkxmlive.sourceforge.net/index.html>

¹⁶<http://docbook.sourceforge.net/projects/dsssl/>

¹⁷<http://sources.redhat.com/docbook-tools/>

можно также пользоваться языком *XSLT*¹⁸ — *XSL Transformations*. В *Sisyphus XSL-таблицы стилей для Docbook/XML*¹⁹ находятся в пакете `docbook-style-xsl`.

*Процессор*²⁰, обрабатывающий XSL-таблицы стилей, находится в *Sisyphus* в пакете `xsltproc`.

Пример 3.2. Преобразование XML-документа в формат HTML с использованием XSL-стилей

```
xsltproc --xinclude \  
--stringparam base.dir html/ \  
--stringparam chunker.output.encoding "KOI8-R" \  
/usr/share/xml/docbook/xsl-stylesheets/html/chunk.xsl file.xml
```

В этом примере используется стиль *chunk*, который создаёт отдельные html-файлы для каждого раздела в подкаталоге `html`. Параметр `chunker.output.encoding` указывает кодировку выходных файлов.

Подсказка

При использовании стиля *docbook* на выходе получится один большой html-файл со всем содержимым. Параметр `chunker.output.encoding` при этом не учитывается и в документе вместо букв будут их номера в *UNICODE*. Такие документы корректно отображаются браузерами, просто занимают больше места, чем документы с кодировкой, например, `koi8-r`.

Опытным путем выяснено, что использование XSL-стилей даёт лучшие результаты, чем использование DSSSL-стилей.

Оформление

Обязательные требования к оформлению документов:

- Документы должны создаваться в формате *DocBook/XML* актуальной версии (в данный момент используется версия 4.2);

¹⁸<http://www.rol.ru/news/it/helpdesk/xslt01.htm>

¹⁹<http://docbook.sourceforge.net/projects/xsl/>

²⁰<http://xmlsoft.org/XSLT/>

- Документы должны соответствовать правилам XML и указанного DTD, проверить это можно командой

```
xmllint --noout --noent --xinclude --postvalid file.xml
```

В CVS эту команду выполняет цель *check* (**make check**). Указанные аргументы отключают вывод содержимого документа, заменяют сущности (*entities*) их значениями, включают обработку включений (*XML Inclusions*²¹), и после всего этого проверяют валидность документа;

- Должны быть заполнены теги **author** (автор), **revhistory** (список ревизий, в каждой должны быть номер, дата и описание), **abstract** (аннотация). Дату желательно указывать на английском языке, пометив тег **date** атрибутом **lang="en"**;
- Крайне желательно использовать теги **section** вместо **sect1**, **sect2** и т.д. — это упрощает изменения документа;
- В корневом теге документа должен быть указан атрибут **lang**, указывающий язык документа. Для русского языка это **lang="ru"**;
- При написании документов нужно использовать букву ё, её замена на **e** недопустима;
- Если в создании документа принимал участие кто-либо кроме автора (например, оформитель документа в *DocBook*), должен быть заполнен тег **othercredit** с описанием участия;
- Для часто повторяющихся конструкций или терминов нужно использовать сущности (*entities*);
- Документы могут создаваться в любой кодировке, поддерживаемой стандартом XML, при этом нужно указывать «официальное» название кодировки. Например, *windows-1251*, а не *cp1251*;
- В документах нельзя использовать символы, которых нет в стандарте языка XML, например, кавычки-ёлочки, длинное тире и т.д., пользуйтесь определениями символов (*character entities*) для их указания. Например, нужно указывать сущность **—** для указания длинного тире (-);
- В документах не должно быть явного форматирования, например, выделений текста символами кавычек или знаков **<**; — пользуйтесь соответствующими тегами, например, для кавычек тегом **quote**, для указания тега достаточно просто **sgmltag**;
- Желательно следить за тем, чтобы в теги попадали те слова, которые к эти тегам относятся, например, не нужно писать внутри тегов точку (в конце предложения);

²¹<http://www.w3.org/TR/xinclude/>

- В заголовках статей и разделов точки в конце ставить не нужно;
- В списках в конце перечислений нужно ставить точку с запятой;

Помещение в CVS

В CVS документы находятся в отдельных каталогах (модулях), каждый из которых закреплен за своим *мантейнером*.

В каждом каталоге может быть только один основной документ. XML-файлов может быть несколько, например, для включения их в основной документ.

Если нужно описать две версии одной программы, можно либо объединить описания в один документ, либо сделать два разных документа, если описания сильно отличаются. Если описания взаимоисключающие, нужно делать их в одном документе, используя Profiling. Название основного (корневого) документа в модуле должно совпадать с названием модуля — это нужно для объединения документов. Желательно давать каталогам с документами названия в нижнем регистре.

Для того, чтобы документ из каталога был включен в автоматическое объединение документов для какого-либо раздела (`alt`, `admin` и т.д.), в каталоге должен быть пустой файл `public-document`.

В объединяющих документах не должно быть собственного содержания, только указание включений — это облегчает редактирование документов.

Замечание

В CVS каталоги с документами *должны* повторять структуру элементов *DocBook*. Помещение новых документов в определённые каталоги должно быть оговорено в списке рассылки `<docs@altlinux.ru>`.

В CVS хранятся только исходные тексты документов и сопровождающие их файлы модуля, например, `Makefile`, `README` и `TODD`. Производные форматы, например, HTML, в CVS недопустимы. Желательно указывать кодировку в сопровождающих файлах, если они написаны простым текстом на русском языке.

В каждом модуле должен находиться файл `Makefile`, образец такого файла можно получить в CVS, в каталоге `$CVSROOT/docs/doc-template`. `Makefile` состоит из включаемых файлов с целями, которые необязательны для включения, но удобны для централизованного обновления правил.

Подсказка

После копирования шаблонного `Makefile` нужно указать правильное значение относительного пути в `MAKEFILES_DIR`.

Замечание

Файл с целью `dropdtd` из CVS удалён — в данный момент документы объединяются с помощью технологии XML *Inclusions*.

В CVS помещаются только «валидные» документы (не нарушающие правил XML и типа документа). При этом изменения должны быть описаны в самом документе (тег `revision`), кроме записи при команде `cvs commit file.xml`.

Таблица 3.1. Структура каталогов в CVS

<code>user</code>	— документы для пользователя	
	<code>multimedia</code>	
		<code>document1</code>
		<code>document2</code>
	<code>network</code>	
		<code>document1</code>
		<code>document2</code>
	<code>office</code>	
		<code>document1</code>
	<code>publishing</code>	
		<code>document1</code>
	<code>other</code>	
		<code>document1</code>
<code>admin</code>	— документы для администратора	
	<code>network</code>	
		<code>document1</code>
		<code>document2</code>

Таблица 3.2. Элементы *DocBook*, соответствующие каталогам в CVS

<book>		
	<part>	
		<article>
		<chapter>
	<part>	
		<article>

Образец документа в *DocBook*

```

<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd"
[
<!ENTITY % AltCommonEntities PUBLIC
"-//ALTLINUX//ENTITIES Common Documentation//RU"
"http://docs.altlinux.ru/ru/entities/common.ent">
%AltCommonEntities;

<!ENTITY BASEID "doc-template">

]>

<chapter id="&BASEID;" lang="ru">

  <chapterinfo>
    <revhistory>
      <revision>
        <revnumber>0.3</revnumber>
        <date lang="en">29 Nov 2002</date>
        <revremark>
          Тип документа изменён с article на section.
        </revremark>
      </revision>
      <revision>
        <revnumber>0.2</revnumber>
        <date lang="en">22 Oct 2002</date>
        <revremark>
          Кодировка изменена с cp1251 на windows-1251. Изменён
          почтовый адрес автора. Добавлен тег othercredit с указанием
          Антона Бояршинова. Теги sect1 изменены на section.
        </revremark>
      </revision>
    </revhistory>
  </chapterinfo>
</chapter>

```

```

</revremark>
  </revision>
  <revision>
<revnumber>0.1</revnumber>
<date lang="en">28 May 2002</date>
<revremark>
  Начальный вариант
</revremark>
  </revision>
</revhistory>
<author lang="en">
  <firstname>Vitaly</firstname>
  <surname>Ostanin</surname>
  <affiliation>
<address>
  <email>vyt@altlinux.ru</email>
  <otheraddr>JID: vyt@vzljot.ru</otheraddr>
</address>
  </affiliation>
</author>
<othercredit lang="en">
  <firstname>Anton</firstname>
  <surname>Boyarshinov</surname>
  <affiliation>
<address>
  <email>boyarsh@ru.echo.fr</email>
</address>
  </affiliation>
  <contrib lang="ru">
Перевод документа на использование <emphasis>PUBLIC
Entities</emphasis>
  </contrib>
</othercredit>
</chapterinfo>

<title>Шаблон статьи для &ALTDOS;</title>

<section>
  <title>Раздел</title>
  <para>
    Текст раздела.
  </para>
</section>

</chapter>

```

этим шаблоном статьи для *ALT Linux Team Documentation* и шаблонами сопровождающих файлов.

Ссылки между документами

В каждом документе, который использует ссылки (исключая ссылки на ресурсы URL), должна быть определена сущность `&BASEID;`. Значение `&BASEID;` должно совпадать с расположением документа в иерархии документов. Например, для этого документа `&BASEID;` равно `alt.devel.docs-howto`.

Для ссылок (в общем случае) используется атрибут `id`, который должен быть уникален в пределах всего документа. При объединении документов он должен быть уникален в пределах всех документов, поэтому его значение должно начинаться с сущности `&BASEID;`, например, `id=&BASEID;` для корневого тега документа, или `id=&BASEID;.intro` для раздела со вступлением.

Подсказка

Атрибут `id` входит в набор общих атрибутов *DocBook* и его можно указывать для любого тега.

Для создания ссылки внутри одного документа можно воспользоваться несколькими тегами:

- | | |
|--------------------|---|
| <code>xref</code> | Ссылка с пустым содержимым. Текст ссылки генерируется из того места, куда она указывает. Удобно использовать для ссылок на разделы, не указывая их название — оно будет подставлено в текст ссылки автоматически. |
| <code>link</code> | Обычная ссылка на идентификатор в документе, может содержать текст. |
| <code>ulink</code> | Ссылка на сетевой ресурс (например, на <i>web</i> -сайт), заданный с помощью URL. |
| <code>olink</code> | Ссылка на место в другом XML документе. Может быть пустой, тогда текст генерируется из места назначения. На этапе создания такой ссылки ещё неизвестно, как будет сформирован вывод другого документа (и куда она должна в итоге указывать), поэтому ссылка указывается не напрямую, а с дополнительным именем документа. Для того, чтобы такие ссылки работали, должна создаваться база ссылок для каждого возможного выходного формата. |

Пример 3.3. Ссылка на другой XML документ

```
<olink targetdoc="docs-howto"↵
targetptr="alt.devel.docs-howto.welcome">
Добро пожаловать!
</olink>
```

В этом примере ссылка указывает на идентификатор `alt.devel.docs-howto.welcome` в документе `docs-howto`. При обработке документа ссылка разрешается с помощью общей базы ссылок (указывается параметром для XSL-стилей).

Замечание

В оригинальной документации *DocBook* находится устаревшее описание `olink`. Хорошее описание ссылок между документами можно найти в *документации*²² Роберта Стейтона.

Иллюстрации

Иллюстрации к документам должны создаваться в формате PNG. При сохранении картинки с экрана текущий видеорежим должен быть *24 bpp* (бит на пиксел) — это избавляет от неприятных эффектов при печати, возникающих при передаче цветов чередующимися комбинациями точек разного цвета.

Для хранения файлов с иллюстрациями в каталоге с документом должен быть создан подкаталог `images`.

Все файлы с изображениями должны иметь имена, уникальные в пределах всего набора документов (это нужно для избежания конфликтов при объединении документов). Из-за особенностей обработки файлов в TeX имена файлов с изображениями должны содержать только одну точку — перед расширением. Поэтому названия файлов должны состоять из полного пути к документу, имени документа, названия изображения и расширения. В качестве разделителя при сборе полного имени должен использоваться дефис, перед расширением ставится точка.

Например, файл в формате PNG с изображением главного окна программы (*main-window*) для документа `user/network/instant-messaging/jabber/psi/psi.xml` должен лежать в каталоге `user/network/instant-messaging/jabber/psi/images` и называться `user-network-instant-messaging-jabber-psi-main-window.png`

Для удобства ссылок на изображения удобно использовать сущность `&BASEIMAGES;` со значением, например, `images/user-network-instant-messaging-jabber-psi`.

²²<http://www.sagehill.net/xml/docbookxsl/0linking.html>

Для удобства указания «плавающих» изображений нужно указывать их внутри тега `figure`, в котором атрибут `float` определяет, может изображение «плавать» (`float=1`), или нет (`float=0`).

Пример 3.4. Включение «плавающей» иллюстрации

```
<figure float="1">
<title>Выбор профиля</title>
<screenshot>
  <graphic fileref="&BASEIMAGES;-open-profile.png"/>
  </screenshot>
</figure>
```

Информация о *DocBook*

DocBook — это описание типа документа (DTD) для языка XML, разработанное для создания документации.

Ссылки на информацию по XML (на русском языке):

- *XML в 10 тезисах*²³;
- *Русские переводы*²⁴ документов *World Wide Web Consortium*²⁵ (W3C);
- *Переводы*²⁶ В. Ярошевича статей об XML и не только;
- Новости XML и ресурсы для разработчиков на *xmlhack.ru*²⁷;
- *Страница SGML*²⁸ Бориса Тоботраса. SGML — близкий родственник XML, поэтому многое на этой странице применимо и к XML. Интересна статья *SGML: с чем это едят?*²⁹;
- *Построение системы XML/XSL-преобразований*³⁰;
- *XSL-преобразования*³¹;

²³<http://croll.hotbox.ru/W3C/XML/1999/XML-in-10-points/>

²⁴<http://croll.hotbox.ru/W3C/Consortium/Translation/russian-ru.html>

²⁵<http://www.w3.org>

²⁶<http://www.raleigh.ru/XML/>

²⁷<http://www.xmlhack.ru/>

²⁸<http://xtalk.opensource.ru/SGML/>

²⁹<http://xtalk.opensource.ru/SGML/SGML-article.html>

³⁰<http://www.webmascon.com/technologies/6a.asp>

³¹http://www.xml.nsu.ru/extra/xslt_0.xml

Книги по XML (на русском языке):

- *Эрик Рэй* «Изучаем XML», оригинал издательства *O'Reilly*³², перевод *С. Маккавеева*, издательство *Символ-Плюс*³³, ISBN 5-93286-023-5;
- *Н. Пити-Моултис*, *Ч. Курк* «XML Black Book», оригинал издательства «The Coriolis Group», перевод издательства *BHV* — *Санкт-Петербург*³⁴, ISBN 5-7791-0112-4;

Ссылки на информацию по XML (на английском языке):

- *Документы по XML на W3C*³⁵. Свободно владеющим английским языком тут раздолье, поэтому больше ссылок можно не указывать :)

Ссылки на информацию по *Docbook* (на русском языке):

- *Перевод статьи*³⁶ «Написание документации, часть III: *DocBook/XML*» на *сайте*³⁷ русской версии «Linux Gazette»;

Ссылки на информацию по *Docbook* (на английском языке):

- Домашняя страница *Docbook*;
- *DocBook: The Definitive Guide*³⁸. Можно скачать *одним файлом*³⁹;
- *Docbook FAQ*⁴⁰;
- *Документация*⁴¹ Роберта Стейтона (Robert Stayton) об использовании стилей, модульных документов и ссылок между ними.

³²<http://www.oreilly.com/>

³³<http://www.symbol.ru>

³⁴<http://www.bhv.ru>

³⁵<http://www.w3.org/XML/>

³⁶<http://gazette.linux.ru.net/lg75/articles/rus-spiel.html>

³⁷<http://gazette.linux.ru.net/>

³⁸<http://docbook.org/tdg/en/html/docbook.html>

³⁹<http://docbook.org/tdg/en/tdg-en-html-2.0.7.zip>

⁴⁰<http://www.dpawson.co.uk/docbook/index.html>

⁴¹<http://www.sagehill.net/xml/docbookxsl/index.html>

Добро пожаловать!

Если вы хотите внести свой вклад в создание документации, присоединяйтесь — для этого нужно написать на адрес <join@altlinux.ru>. Желательно указать в письме, какие документы вы хотите написать, вычитать или поддерживать. После этого вы будете подписаны на список рассылки <docs@altlinux.ru>.

Для доступа к CVS с документацией нужно пройти процедуру регистрации, она состоит в генерации ключа *OpenSSH* (алгоритм DSA, длина ключа 2048 бит) и отсылке его публичной части на адрес <join@altlinux.ru>. Подробная информация о генерации ключей доступна на странице руководства по ssh-keygen (`man ssh-keygen`). Программы ssh-keygen и ssh находятся в *Sisyphus* в пакетах `openssh-clients` и `openssh`.

Пример 3.5. Генерация новой пары ключей SSH

```
ssh-keygen -t dsa -b 2048
```

Пример 3.6. Пример конфигурации в `~/.ssh/config` для доступа к *devel*:

```
Host devel
  HostName devel.altlinux.ru
  User alt_user
  Protocol 2
  ForwardX11 no
  ForwardAgent no
  Compression no
```

Замечание

Разработчики, уже имеющие доступ к *devel*, имеют также доступ на чтение к CVS с документацией.

Можно сразу создать пару ключей GPG (*PGP*). Подробная информация о программе GnuPG доступна в *Sisyphus* в пакете `gnupg-manual`.

Пример 3.7. Генерация новой пары ключей GPG в интерактивном режиме

```
gpg --gen-key
```

После получения доступа к *devel* можно получить файлы из CVS.

Пример 3.8. Получение файлов из CVS

```
export CVS_RSH=ssh; export CVSROOT=devel:/cvs/docs; cvs co -P docs
```

Более подробная информация о работе с CVS доступна на сайте *cvs.ru*⁴².

О проблемах с доступом пишите в список рассылки `<docs@altlinux.ru>`.

⁴²<http://www.cvs.ru>