

ebuild(5), Gentoo Linux

ИМЯ

ebuild - внутренний формат, переменные и функции в сценарии **ebuild**

ОПИСАНИЕ

Программа **ebuild(1)** воспринимает в качестве аргумента единственный сценарий ebuild. Этот сценарий содержит переменные и команды, указывающие, как загружать, распаковывать, вносить исправления, компилировать, устанавливать и включать в дерево пакет из его оригинальных исходных текстов. Кроме всего этого, сценарий ebuild может также содержать команды, которые необходимо выполнять до/после установки/удаления пакета.

ПРИМЕРЫ

Вот простой пример сценария ebuild:

```
# Copyright 1999-2002 Gentoo Technologies, Inc.
# Distributed under the terms of the GNU General Public License v2
# $Header: $
DESCRIPTION="Super-useful stream editor (sed)"
SRC_URI="ftp://alpha.gnu.org/pub/gnu/sed/${P}.tar.gz"
HOMEPAGE="http://www.gnu.org/software/sed/sed.html"
KEYWORDS="~x86"
SLOT="0"
LICENSE="GPL-2"
IUSE=""
DEPEND="virtual/glibc"
RDEPEND="virtual/glibc"

src_compile() {
    econf || die "could not configure"
    emake || die "emake failed"
}

src_install() {
    into /usr
    doinfo doc/sed.info
    doman doc/sed.1
    into /
    dobin sed/sed
    dodir /usr/bin
    dosym /bin/sed /usr/bin/sed
    dodoc COPYING NEWS README* THANKS TODO AUTHORS BUGS ANNOUNCE
}
```

ПЕРЕМЕННЫЕ

ПРИМЕЧАНИЯ ОБ ИСПОЛЬЗОВАНИИ

- Переменные **PORTAGE*** и **PORTDIR*** можно найти в файле **make.conf(5)**.

- При присвоении значений переменным в пакетах ebuild нельзя указывать пробел между именем переменной и знаком равенства.

P

Эта переменная содержит имя пакета без номера редакции ebuild. Эту переменную нельзя менять НИКОГДА.

```
xfree-4.2.1-r2.ebuild --> $P=='xfree-4.2.1'
```

PN

Содержит имя сценария без номера версии.

```
xfree-4.2.1-r2.ebuild --> $PN=='xfree'
```

PV

Содержит номер версии без номера редакции.

```
xfree-4.2.1-r2.ebuild --> $PV=='4.2.1'
```

PR

Содержит номер редакции (revision number) или 'r0', если номер редакции не задан.

```
xfree-4.2.1-r2.ebuild --> $PR=='r2'
```

PVR

Содержит номер версии с номером редакции.

```
xfree-4.2.1-r2.ebuild --> $PVR=='4.2.1-r2'
```

PF

Содержит полное имя пакета: [PN]-[PV]-r[PR]

```
xfree-4.2.1-r2.ebuild --> $PF=='xfree-4.2.1-r2'
```

A

Содержит все исходные файлы, необходимые для пакета. Эта переменная не должна определяться. Она генерируется автоматически на основе переменных [SRC_URI](#).

WORKDIR = "\${PORTAGE_TMPDIR}/portage/\${PF}/work"

Содержит полное имя корневого каталога построения пакета. Не изменяйте эту переменную.

FILESDIR = "\${PORTDIR}/\${CATEGORY}/\${PN}/files"

Содержит полное имя подкаталога 'files' в соответствующем каталоге пакета в дереве портежей. Не изменяйте эту переменную.

S = "\${WORKDIR}/\${P}"

Содержит полное имя временного каталога для процесса построения (temporary build directory). Эта переменная используется функциями [src_compile](#) и [src_install](#). Обе они выполняются в каталоге S. Эта переменная может меняться в соответствии с каталогом, в который распаковывается архив (tarball) пакета.

T = "\${PORTAGE_TMPDIR}/portage/\${PF}/temp"

Содержит полное имя временного каталога. Его можно использовать для любых целей.

D = "\${PORTAGE_TMPDIR}/portage/\${PF}/image"

Содержит полное имя временного каталога для установки. Каждая операция записи, выполняемая не через вспомогательные средства и функции (представленные далее), должна выполняться в каталоге \${D}. Не изменяйте значение этой переменной.

DESCRIPTION = "A happy little package"

Эта переменная должна содержать краткое описание пакета.

SRC_URI = "http://happy.com/little/\${P}.tar.gz"

Содержит список адресов URI для соответствующих исходных файлов. Она может содержать несколько адресов URI для одного исходного файла. Если файла нет на [GENTOO_MIRROR](#), выбирается самый быстрый из перечисленных сайтов.

HOMEPAGE = "http://happy.com/"

Должна содержать список адресов URL для основных сайтов с исходными текстами и другой информацией для пакета.

KEYWORDS = [-][x86,ppc,sparc,mips,alpha,arm,hppa]

Должна содержать соответствующий список архитектур, на которых пакет ebuild определенно работает/не работает. По умолчанию, если неизвестно, работает ли пакет ebuild на определенной архитектуре, соответствующее ключевое слово (KEYWORD) просто не указывается. Если пакет ebuild не будет работать на определенной архитектуре, укажите ее с минусом, например, -ppc. Если

пакет ebuild посылается для включения, у него должны быть установлены ключевые слова `~arch` для архитектур, на которых он ПРЕДПОЛОЖИТЕЛЬНО РАБОТАЕТ (PROVEN TO WORK). Пакеты с соответствующими ключевыми словами можно будет демаскировать для тестирования путем установки `ACCEPT_KEYWORDS="~arch"` в командной строке или в файле `make.conf(5)`. Полный список имен архитектур см. в файле `/usr/portage/profiles/arch.list`.

SLOT

Устанавливает слот (SLOT) для пакетов, которым необходимо сосуществовать. По умолчанию, необходимо устанавливать `SLOT="0"`, если только вы не уверены, что явно нужно другое значение. Эта переменная НИКОГДА не должна оставаться неопределенной.

LICENSE

Значение должно представлять собой список (через пробел) имен лицензий, по которым распространяется пакет. Это **должны** быть лицензии, представленные в каталоге `/usr/portage/licenses/`. Если соответствующей лицензии в системе портежей нет, ее необходимо явно добавить.

IUSE

Значением должен быть список всех необходимых флагов `USE`, использующихся в сценарии построения пакета. В этой переменной не должны указываться только флаги `USE`, задающие архитектуру (см. [KEYWORDS](#)).

DEPEND

Эта переменная должна содержать список всех пакетов, которые необходимы для компиляции программы.

Атомы зависимости

Атом зависимости - это просто зависимость, используемая системой портежей при определении взаимосвязей между пакетами. Учтите, пожалуйста, что если атом еще не был включен в дерево пакетов, включаться будет самая последняя доступная версия.

Базовые атомы

Базовый атом - это просто полное имя категории/имя пакета. Вот примеры базовых атомов:

```
sys-apps/sed  
sys-libs/zlib  
net-misc/dhcp
```

Версии атомов

Иногда необходимо дополнительное уточнение, что подойдут только определенные версии атомов. Учтите, что версии необходимо указывать для префикса - базового атома. Таким образом, номер версии просто добавляется как суффикс к базовому атому:

```
sys-apps/sed-4.0.5  
sys-libs/zlib-1.1.4-r1  
net-misc/dhcp-3.0_p2
```

Номера версий обычно состоят из двух или трех чисел, разделенных точками, например, **1.2** или **4.5.2**. После этой строки может идти буква, например, **1.2a** или **4.5.2z**. Учтите, что эта буква не должна обозначать статус **alpha**, **beta** и т.д. Для этого используется необязательный суффикс: **_alpha**, **_beta**, **_pre** (пре-релиз), **_rc** (релиз-кандидат) или **_p** (patch). Это означает, что для третьего пре-релиза пакета надо будет использовать версию типа **1.2_pre3**.

Префиксные операторы атома [`> >= >= >`]

Иногда необходимо указывать зависимость от группы версий, не задавая постоянно каждую из них явно. Для этого предоставляются стандартные булевые операторы:

```
>media-libs/libgd-1.6  
>=media-libs/libgd-1.6  
=media-libs/libgd-1.6  
>=media-libs/libgd-1.6  
>media-libs/libgd-1.6
```

Дополнительные префиксы [!~] и суффиксы [*] атомов

Чтобы стало еще интереснее, обеспечивается возможность указывать блокирующие пакеты и соответствующие диапазоны версий. Учтите также, что эти дополнительные префиксы/суффиксы можно сочетать любым способом с представленными выше классами атомов. Вот ряд типичных примеров, которые можно найти в дереве портежей:

```
!app-text/dos2unix  
=dev-libs/glib-2*  
!=net-fs/samba-2*  
~net-libs/libnet-1.0.2a
```

! означает блокировку одновременной установки пакетов.

- * означает соответствие любой версии пакета с указанной базой. Поэтому версия '2*' будет соответствовать '2.1', '2.2' и т.д.
- ~ означает соответствие любой редакции указанной базовой версии. Поэтому в представленном выше примере будут соответствовать версии '1.0.2a', '1.0.2a-r1', '1.0.2a-r2' и т.д.

Динамические зависимости

Иногда программы могут зависеть от разных пакетов и версий, в зависимости от значения переменной **USE**. Система портежей предлагает несколько опций для учета этого. Учтите, что при использовании следующих синтаксических конструкций, каждая конструкция рассматривается как 1 атом в соответствующем контексте. Это означает, что каждый атом может включать за счет задания условий несколько атомов, иложенность эта может быть любой глубины.

<переменная USE>? (<атом зависимости>)

Для включения библиотеки **jpeg** при указании пользователем **jpeg** в переменной USE, используйте следующую конструкцию:

```
jpeg? ( media-libs/jpeg )
```

<переменная USE>? (<атом если истина>) : (<атом если ложь>)

Работает аналогично терциарному оператору **:** языка C. Если пакет использует **GTK2** или **GTK1**, но не оба вместе, это можно выразить так:

```
gtk2? ( =x11-libs/gtk+-2* ) : ( =x11-libs/gtk+-1* )
```

В результате, по умолчанию будет использоваться более новая библиотека **GTK2**.

!<переменная USE>? (<атом>)

Если необходимо включать пакет только при отсутствии определенной опции в переменной USE, используется следующая конструкция:

```
!nophysfs? ( dev-games/physfs )
```

Она часто используется, когда необходимо добавить необязательную поддержку той или иной возможности, и установливать ее по умолчанию.

|| (<атом> <атом> ...)

Когда пакет может работать с несколькими различными пакетами, но категория **virtual** не подходит, можно запросто использовать эту конструкцию.

```
||  
(  
app-games/unreal-tournament  
app-games/unreal-tournament-goty  
)
```

Здесь мы видим, что пакет **unreal-tournament** бывает в обычной версии и в версии **goty**. Поскольку обе они предоставляют один и тот же базовый набор файлов, другой пакет может использовать любой из них. Добавление категории **virtual** не подходит, поскольку она не охватывает соответствующие файлы.

Еще один хороший пример - это когда пакет может строиться с несколькими видеокартами, но для каждого конкретного построения должен использоваться какой-то один из них.

```
||  
(  
sdl? ( media-libs/libsdl )  
svga? ( media-libs/svgalib )  
opengl? ( virtual/opengl )  
ggi? ( media-libs/libggi )  
virtual/x11  
)
```

В данном случае, будет выбран только один из пакетов, а порядок предпочтения определяется порядком их перечисления. Поэтому библиотека **sdl** является наиболее вероятным кандидатом, далее идет **svga**, затем - **opengl**, **ggi** и по умолчанию, если пользователь не задаст явно одну из предыдущих библиотек, будет использоваться **X**.

RDEPEND

Эта переменная должна содержать список всех пакетов, необходимых для работы данной программы (другими словами, это зависимости *времени выполнения*). Если она не установлена, по умолчанию используется значение переменной **DEPEND**.

Для задания динамических зависимостей можно использовать такие же конструкции, как и для переменной **DEPEND**.

PDEPEND

Эта переменная должна содержать список всех пакетов, которые необходимо установить после включения в дерево данной программы.

Для задания динамических зависимостей можно использовать такие же конструкции, как и для переменной **DEPEND**.

RESTRICT = [nostrip,nomirror,fetch,nouserpriv]

Задает список (через пробел) ограничений возможностей системы портежей.

nostrip

Из результирующих двоичных модулей/библиотек не будет удаляться отладочная информация.

nouserpriv

Отключает **userpriv** для определенных пакетов.

nomirror

Файлы, указанные в переменной **SRC_URI**, не будут загружаться с зеркальных сайтов, **GENTOO_MIRRORS**.

fetch

Аналогично **nomirror**, но файлы не будут выбираться и через переменную **SRC_URI**.

PROVIDE = "virtual/TARGET"

Эта переменная должна использоваться только если пакет предоставляет виртуальную цель (из категории **virtual**). Например, **blackdown-jdk** и **sun-jdk** предоставляют **virtual/jdk**. Это позволяет пакетам задавать зависимости от **virtual/jdk**, а не явно от реализации **blackdown** или **sun**.

ФУНКЦИИ

pkg_nofetch

Если **fetch** указано в переменной **RESTRICT**, эта функция будет вызываться, когда не удастся найти файлы в **SRC_URI**. Пригодится для выдачи информации пользователю о том, как получить соответствующие файлы. Достаточно выдать сообщение и нормально завершить работу функции. Не завершайте функцию вызовом **die**.

pkg_setup

Эту функцию можно использовать, если для пакета необходимо выполнить определенные действия по настройке или проверки, прежде чем делать все остальное.

Исходным рабочим каталогом для этой функции будет **\${PORTAGE_TMPDIR}**.

src_unpack

Эта функция используется для распаковки всех исходных текстов, указанных в **A** в каталог **WORKDIR**. Если эта функция в сценарии ebuild не определена, вызывается **unpack \${A}**. В этой функции необходимо устанавливать патчи и выполнять все остальные изменения перед конфигурированием/компиляцией.

Исходным рабочим каталогом для этой функции будет **\${WORKDIR}**.

src_compile

В этой функции должны выполняться все необходимые шаги конфигурирования и компиляции.

Исходным рабочим каталогом для этой функции будет **\${S}**.

src_install

Должна содержать все необходимое для установки пакета во временном каталоге установки.

Исходным рабочим каталогом для этой функции будет **\${S}**.

pkg_preinst **pkg_postinst**

В этих функциях должны выполняться все модификации в "живой" файловой системы, необходимые до и после включения пакета в дерево. В **pkg_postinst** также необходимо указывать комментарий для пользователя, поскольку он будет выводиться напоследок.

Исходным рабочим каталогом для этой функции будет **\$PWD**.

pkg_prerm pkg_postrm

Аналогично функциям **pkg_*inst**, но - для исключения пакета из дерева (unmerge).

Исходным рабочим каталогом для этой функции будет **\$PWD**.

config

Эта функция должна содержать необязательные основные шаги по конфигурированию.

Исходным рабочим каталогом для этой функции будет **\$PWD**.

ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ: ОБЩИЕ

die [<причина>]

Вызывает прекращение работы текущего процесса **emerge**. Выводимая при этом информация будет включать **причину**.

use <элемент USE>

Если элемент **USE** указан в переменной **USE**, элемент **USE** будет выведен, и функция вернет 0. Если же элемент **USE** в переменной **USE** не указан, функция вернет 1.

Пример:

```
if [ `use gnome` ] ; then
    guiconf="--enable-gui=gnome --with-x"
elif [ `use gtk` ] ; then
    guiconf="--enable-gui=gtk --with-x"
elif [ `use X` ] ; then
    guiconf="--enable-gui=athena --with-x"
else
    # Версия с графическим интерфейсом строиться не будет
    guiconf=""
fi
```

use_with <элемент USE> [<опция configure>]

Пригодится для создания специфических опций, передаваемых сценарию **configure**. Если элемент **USE** указан в переменной **USE**, будет выведена строка **--with-[<опция configure>]**. Если же элемент **USE** не указан в переменной **USE**, будет выведена строка **--without-[<опция configure>]**. Если **опция configure** не указана, вместо нее будет использован элемент **USE**.

Пример:

```
USE="jpeg"
myconf=`use_with jpeg libjpeg`
(myconf теперь имеет значение "--with-libjpeg")
USE=""
myconf=`use_with jpeg libjpeg`
(myconf теперь имеет значение "--without-libjpeg")
USE="pic"
myconf=`use_with pic`
(myconf теперь имеет значение "--with-pic")
```

use_enable <элемент USE> [<опция configure>]

Пригодится для создания специфических опций, передаваемых сценарию **configure**. Если элемент **USE** указан в переменной **USE**, будет выводиться строка **--enable-[<опция configure>]**. Если же элемент **USE** в переменной **USE** не указан, будет выводиться строка **--disable-[<опция configure>]**. Если **опция configure** не указана, вместо нее будет использоваться сам элемент **USE**.

Пример см. в описании функции [use_with](#).

has <элемент> <список элементов>

Если **элемент** входит в **список элементов**, такой **элемент** выводится и функция возвращает 0. В противном случае, ничего не выводится и функция возвращает 1.

Разделитель **списка элементов** задается переменной **IFS**. Ее стандартное значение - ' ', пробел. Это - переменная командного интерпретатора [bash\(1\)](#).

has_version <категория>/<пакет>-<версия>

Проверяет, установлена ли **категория/пакет-версия** в системе. Параметр может иметь любые значения, допустимые для переменной [DEPEND](#). Эта функция возвращает 0, если **категория/пакет-версия** установлена и 1 в противном случае.

best_version <имя пакета>

Эта функция будет искать **имя пакета** в базе данных установленных в настоящее время программ и выводить "лучшую версию" уже установленного пакета. Функция возвращает 0, если в системе есть пакет с соответствующим именем. В противном случае она возвращает 1.

Пример:

```
VERINS=`best_version net-ftp/glftpd`
```

Переменная **VERINS** теперь имеет значение "**net-ftp/glftpd-1.27**", если в системе установлен **glftpd-1.27**.

ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ: ВЫВОД

einfo "информационное сообщение"

Если необходимо вывести сообщение пользователю для прочтения, используйте функцию **einfo**. Она работает аналогично [echo\(1\)](#), но добавляет дополнительную информацию, привлекающую внимание пользователя.

ewarn "предупреждающее сообщение"

Аналогична **einfo**, но должна использоваться для предупреждения пользователя.

error "сообщение об ошибке"

Аналогична **einfo**, но должна использоваться для выдачи пользователю сообщения об ошибках.

ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ: РАСПАКОВКА

uppack <исходник> [<список дополнительных исходников>]

Эта функция распаковывает (uncompresses) и/или разархивирует (untars) **список исходников** в текущий каталог. Она добавляет **исходник** к значению переменной [DISTDIR](#).

ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ: КОМПИЛЯЦИЯ

econf [<опции configure>]

Эта функция используется вместо **configure**. Выполняет:

```
configure \
--prefix=/usr \
--host=${CHOST} \
--mandir=/usr/share/man \
--infodir=/usr/share/info \
--datadir=/usr/share \
--sysconfdir=/etc \
--localstatedir=/var/lib \
${EXTRA_ECONF} \
<опции configure>
```

Поэтому можно передавать опции **econf** либо в командной строке, либо через переменную **EXTRA_ECONF**. При явном вызове функции **econf** имеет смысл передавать дополнительные аргументы **econf**, а не использовать переменную **EXTRA_ECONF**.

emake [<опции make>]

Эта функция используется вместо **make**. Добавляет стандартную опцию **MAKEOPTS="-j2"**.
ПРЕДУПРЕЖДЕНИЕ

Если вы собираетесь использовать **emake**, убедитесь что построение возможно при распараллеливании (**make -j2**). Это необходимо тщательно протестировать, поскольку распараллеливание построения **иногда**, хотя и не всегда, приводит к ошибкам.

ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ: УСТАНОВКА

einstall [<опции make>]

Эта функция используется вместо **make install**. Выполняет:

```
make prefix=${D}/usr \
    mandir=${D}/usr/share/man \
    infodir=${D}/usr/share/info \
    datadir=${D}/usr/share \
    sysconfdir=${D}/etc \
    localstatedir=${D}/var/lib \
    <опции make> install
```

Пожалуйста, не используйте эту функцию вместо '**make install DESTDIR=\${D}**'. Это - предпочтительный способ установки пакетов, собираемых с помощью **make** (make-based packages).

prepall prepalldocs prepallinfo prepallman prepallstrip

Эти функции пригодятся, когда пакет устанавливается в каталог **\${D}** с помощью сценариев, например, make-файлов. Если необходимо гарантировать, что библиотеки являются выполняемыми, файлы **aclocal** установлены в нужное место, все файлы **doc/info/man** упакованы, а из выполняемых файлов удалена отладочная информация, используйте этот набор функций.

prepall:

Выполняет **prepallman**, **prepallinfo**, **prepallstrip**, устанавливает для библиотек +x, а затем проверяет каталоги **aclocal**. Обратите внимание, что функция **prepalldocs** при этом **не вызывается**.

prepalldocs:

Упаковывает все **doc**-файлы в каталоге **\${D}/usr/share/doc**.

prepallinfo:

Упаковывает все **info**-файлы в каталоге **\${D}/usr/share/info**.

prepallman:

Упаковывает все **man**-файлы в каталоге **\${D}/usr/share/man**.

prepallstrip:

Удаляет отладочную информацию из всех выполняемых файлов, в том числе, из библиотек.

prepinfo [<каталог>] preplib [<каталог>] preplib.so [<каталог>] prepman [<каталог>] prepstrip [<каталог>]

Аналогичны функциям **prepall**, но немного отличаются.

prepinfo:

Если **каталог** не указан, функция **prepinfo** предполагает использование каталога **usr**. Функция **prepinfo** будет упаковывать все файлы в каталоге **\${D}/dir/info**.

preplib:

Если **каталог** не указан, функция **preplib** предполагает использование каталога **usr**. Функция **preplib** будет выполнять '**ldconfig -n -N**' в каталоге **\${D}/dir/lib**.

preplib.so:

Функция находит все файлы со строкой '.so' в имени в каталоге \${D}/dir и удаляет из них отладочную информацию. Можно указывать несколько каталогов.

prepman:

Если **каталог** не указан, функция **prepman** предполагает использование каталога **usr**. Функция **prepman** будет упаковывать все файлы в каталоге \${D}/dir/man/*/.

prepstrip:

Из всех файлов \${D}/dir удаляется отладочная информация. Можно указывать несколько каталогов.

dopython <команды>

Выполняет команды языка **python** и возвращает результат.

dosed "s:шаблон:замена:g" <имя файла>

Выполняет команду **sed** (с необходимым переименованием/копированием) для указанного **файла**.

Функция 'dosed "s:/usr/local:/usr:g" /usr/bin/some-script' выполняет соответствующую команду замены **sed** для файла \${D}/usr/bin/some-script.

dodir <полное имя>

Создает каталог в каталоге \${D}.

Команда '**dodir /usr/lib/apache**' создает каталог \${D}/usr/lib/apache.

diropts [<опции для install(1)>]

Может использоваться для задания опций утилите **install**, используемой в функции **dodir**. Стандартное значение - **-m0755**.

into <path>

Устанавливает корневой каталог (**DESTTREE**) для других функций типа **dobin**, **dosbin**, **doman**, **doinfo**, **dolib**.

По умолчанию в качестве корневого используется каталог **/usr**.

keepdir <полное имя>

Указывает системе портежей оставлять указанный каталог, даже если он пустой.

В остальном, работает аналогично **dodir**.

dobin <двоичный файл> [<список дополнительных файлов>]

Устанавливает один или все перечисленные двоичные файлы в каталог **DESTTREE/bin**. Создает при этом все необходимые каталоги.

dosbin <двоичный файл> [<список дополнительных файлов>]

Устанавливает один или все перечисленные двоичные файлы в каталог **DESTTREE/sbin**. Создает при этом все необходимые каталоги.

dolib <библиотека> [<список дополнительных библиотек>]**dolib.a <библиотека> [<список дополнительных библиотек>]****dolib.so <библиотека> [<список дополнительных библиотек>]**

Устанавливает библиотеку или все перечисленные библиотеки в каталог **DESTTREE/lib**. Создает при этом все необходимые каталоги.

libopts [<опции для install(1)>]

Может использоваться для передачи опций утилите **install**, используемой в функциях **dolib**.

Стандартное значение - **-m0644**.

doman <man-страница> [<список дополнительных man-страниц>]

Устанавливает страницы справочного руководства в каталоги **DESTDIR/man/man[1-8n]** в зависимости от последних символов в имени файла. Файлы упаковываются с помощью **gzip**, если они еще не упакованы. Функция создает все необходимые каталоги.

dohard <имя файла> <имя связи>

dosym <имя файла> <имя связи>

Выполняет команду **ln** для создания жесткой или символьической связи.

dohtml [-a <типы файлов>] [-r] [-x <список игнорируемых каталогов>] [<список файлов и каталогов>]

Устанавливает указанные в списке через пробел файлы в каталог **/usr/share/doc/\${PF}/html**, если имена этих файлов заканчиваются на **.html**, **.png**, **.js**. Опция **-a** добавляет новые завершения к этому стандартному списку, а опция **-x** позволяет указать каталоги, которые при этом необходимо пропустить (каталог **CVS** исключается по умолчанию). Опция **-r** требует обрабатывать подкаталоги рекурсивно.

doinfo <info-файл> [<список дополнительных info-файлов>]

Устанавливает info-страницы в каталог **DESTDIR/info**. Файлы при этом автоматически упаковываются с помощью **gzip**. Функция создает все необходимые каталоги.

dojar <jar-файл> [<список дополнительных jar-файлов>]

Устанавливает jar-файлы в каталог **/usr/share/\${PN}/lib** и добавляет их в **/usr/share/\${PN}/classpath.env**.

domo <файл локали> [<список дополнительных файлов локали>]

Устанавливает файлы локали в каталог **DESTDIR/usr/share/locale/[LANG]**, в зависимости от завершения имени файла локали. Функция создает все необходимые каталоги.

fowners <владелец> <файл> [<файлы>]

fperms <права> <файл> [<файлы>]

Выполняет **chown** (**fowners**) или **chmod** (**fperms**), устанавливая владельца и права доступа к указанным файлам.

insinto [<полное имя>]

Устанавливает корневой каталог (**INSDESTTREE**) для функции **doins**. По умолчанию в качестве корневого используется каталог **/**.

insopts [<опции для install(1)>]

Может использоваться для задания опций утилите **install**, используемой в функции **doins**. Стандартное значение - **-m0644**.

doins <файл> [<список дополнительных файлов>]

Устанавливает указанные файлы в каталог **INSDESTTREE**. Эта функция использует утилиту **install(1)**.

exeinto [<полное имя>]

Устанавливает корневой каталог (**EXEDESTTREE**) для функции **doexe**. По умолчанию в качестве корневого используется каталог **/**.

exeopts [<опции для install(1)>]

Может использоваться для задания опций утилите **install**, используемой в функции **doexe**. Стандартное значение - **-m0755**.

doexe <выполняемый файл> [<список дополнительных выполняемых файлов>]

Устанавливает указанный выполняемый файл или файлы в каталог **EXEDESTTREE**. Эта функция использует утилиту **install(1)**.

docinto [<полное имя>]

Устанавливает относительный подкаталог (**DOCDESTTREE**), используемый функцией **dodoc**.

dodoc <документ> [<список дополнительных документов>]

Устанавливает один или несколько указанных документов в каталог **/usr/share/doc/\${PF}/DOCDESTTREE**. Файлы автоматически упаковываются утилитой **gzip**. Функция создает все необходимые каталоги.

newbin <старый файл> <новое имя файла>

newsbin <старый файл> <новое имя файла>

newlib <старый файл> <новое имя файла>

newlib.so <старый файл> <новое имя файла>
newlib.a <старый файл> <новое имя файла>
newman <старый файл> <новое имя файла>
newinfo <старый файл> <новое имя файла>
newins <старый файл> <новое имя файла>
newexe <старый файл> <новое имя файла>
newdoc <старый файл> <новое имя файла>

Все эти функции работают аналогично функциям **do***, но работают они только с одним файлом, который устанавливается с **новым именем**.

СООБЩЕНИЯ ОБ ОШИБКАХ

Пожалуйста, сообщайте об ошибках на сайте <http://bugs.gentoo.org/>

ССЫЛКИ

ebuild(1), make.conf(5).

Сценарий **/usr/sbin/ebuild.sh**.

Вспомогательные приложения в каталоге **/usr/lib/portage/bin**.

ФАЙЛЫ

/etc/make.conf

Содержит переменные для процесса построения, имеющие приоритет над заданными в файле **make.defaults**.

/etc/make.globals

Содержит стандартные переменные для процесса построения. Редактировать значения надо не здесь, а в файле **/etc/make.conf**.

АВТОРЫ

Achim Gottinger <achim@gentoo.org>
Mark Guertin <gerk@gentoo.org>
Nicholas Jones <carpaski@gentoo.org>
Mike Frysinger <vapier@gentoo.org>

ЗАГОЛОВОК CVS

\$Header: /home/cvsroot/gentoo-src/portage/man/ebuild.5,v 1.54 2003/10/14 17:29:49 vapier Exp \$

Последнее изменение: февраль 2003 года

Copyleft (no c) - Fuck copyright!, 2003 [B. Кравчук](#), [OpenXS Initiative](#), перевод на русский язык