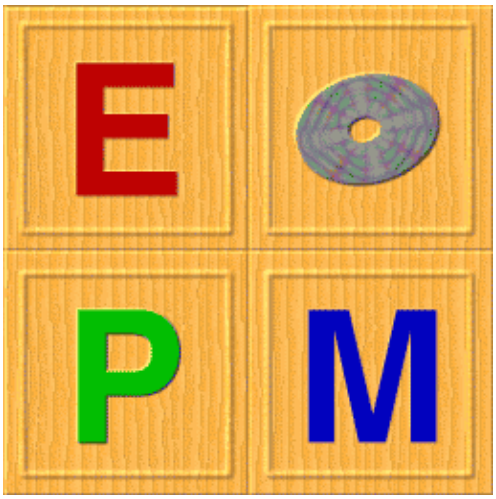

Software Distribution Using the ESP Package Manager

ESP−004−20030723

Michael Sweet

Table of Contents

Table of Contents

Preface

This document provides a tutorial and reference for the ESP Package Manager ("EPM") software, version 3.7.

Organization of this Manual

This document is organized into the following chapters and appendices:

Notation Conventions

Various font and syntax conventions are used in this guide. Examples and their meanings and uses are explained below:

Example	Description
`epm` `epm(1)`	The names of commands; the first mention of a command or function in a chapter is followed by a manual page section number.
/var */usr/bin/epm*	File and directory names.
`Request ID is Printer-123`	Screen output.
`lp -d printer filename ENTER`	Literal user input; special keys like **`ENTER`** are in ALL CAPS.
`foo start of long command \` **`   end of long command ENTER`**	Long commands are broken up on multiple lines using the backslash (\) character. Enter the commands without the backslash.
12.3	Numbers in the text are written using the period (.) to indicate the decimal point.

Abbreviations

The following abbreviations are used throughout this manual:

kb
: Kilobytes, or 1024 bytes

Mb
: Megabytes, or 1048576 bytes

Gb
: Gigabytes, or 1073741824 bytes

Other References

http://www.easysw.com/epm/
: The official home page of the ESP Package Manager software.

http://www.debian.org/devel/
: Debian Developers' Corner

http://techpubs.sgi.com/
: IRIX Documentation On−Line

http://www.rpm.org/
: The RedHat Package Manager home page.

http://docs.sun.com/
: Solaris Documentation On−Line

1 − Introduction to EPM

This chapter provides an introduction to the ESP Package Manager ("EPM").

What is EPM?

Software distribution under UNIX/Linux can be a challenge, especially if you ship software for more than one operating system. Every operating system provides its own software packaging tools and each has unique requirements or implications for the software development environment.

The ESP Package Manager ("EPM") is one solution to this problem. Besides its own "portable" distribution format, EPM also supports the generation of several vendor−specific formats. This allows you to build software distribution files for almost any UNIX/Linux operating system from the same sources.

History and Evolution

When Easy Software Products was founded in 1993, we originally shipped software only for the SGI IRIX operating system. In 1997 we added support for Solaris, which was quickly followed by HP−UX support in 1998.

Each new operating system and supported processor required a new set of packaging files. While this worked, it also meant that we had to keep all of the packaging files synchronized manually. Needless to say, this process was far from perfect and we had more than one distribution that was not identical on all operating systems.

As we began developing the Common UNIX Printing System (http://www.cups.org/) in 1998, our initial goal was to add support for two additional operating systems: Linux and Compaq Tru64 UNIX. If we wanted to avoid the mistakes of the past, we clearly had to change how we produced software distributions.

The first version of EPM was released in 1999 and supported so−called "portable" software distributions that were not tied to any particular operating system or packaging software. Due to popular demand, we added support for vendor−specific packaging formats in the second major release of EPM, allowing the generation of portable or "native" distributions from one program and one set of software distribution files.

Existing Software Packaging Systems

As we looked for a solution to our problem, we naturally investigated the existing open−source packaging systems. Under Linux, we looked at the RedHat Package Manager ("RPM") and Debian packaging software ("dpkg" and "dselect"). For the commercial UNIX's we looked at the vendor−supplied packaging systems. Table 1.1 shows the results of our investigation.

Table 1.1: Software Packaging Formats

Format	Operating Systems[1]	Binaries?	Cross−Platform?	Patches?	Upgrades?	Conflicts?	Requires?	Replaces?	Config Files?	Map Files?
installp	AIX	Yes	No	No	No	Yes	Yes	No	No	No
pkg_add	FreeBSD	Yes	Yes[2]	No	No	No	No	No	No	No
pkg_add	NetBSD OpenBSD	Yes	Yes[2]	No	No	Yes	Yes	No	No	No
dpkg	Corel Linux Debian GNU/Linux	Yes	Yes[2]	No	Yes	Yes	Yes	Yes	Yes	No
swinstall	HP−UX	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes
inst	IRIX	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
pkgadd	Solaris	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes
rpm	Mandrake RedHat SuSE TurboLinux	Yes	Yes[2]	No	Yes	Yes	Yes	No	Yes	No
setld	Tru64 UNIX	Yes	No	No	No	Yes	Yes	No	No	No

1. Standard packaging system for named operating systems.
2. These packaging systems are cross−platform but require the package management utilities to be installed on the platform before installing the package.

As you can see, none of the formats supported every feature we were looking for. One common fault of all these formats is that they do not support a common software specification file format. That is, making a Debian software distribution requires significantly different support files than required for a Solaris pkg distribution. This makes it extremely difficult to manage distributions for multiple operating systems.

All of the package formats support binary distributions. The RPM and Debian formats also support source distributions that specifically allow for recompilation and installation. Only the commercial UNIX formats support patch distributions − you have to completely upgrade a software package with RPM and Debian. All but the Solaris `pkg` format allow you to upgrade a package without removing the old version first.

When building the software packages, RPM and Debian force you to create the actual directories, copy the files to those directories, and set the ownerships and permissions. You essentially are creating a directory for your software that can be archived in the corresponding package format. To ensure that all file permissions and ownerships are correct, you must build the distribution as the root user or use the `fakeroot` software, introducing potential security risks and violating many corporate security policies. It can also make building distributions difficult when dynamic data such as changing data files or databases is involved.

The commercial UNIX formats use software list files that map source files to the correct directories and permissions. This allows for easier delivery of dynamic data, configuration management of what each distribution actually contains, and eliminates security issues with special permissions and building distributions as the root user. Using the proprietary format also has the added benefit of allowing for software patches and using the familiar software installation tools for that operating system. The primary disadvantage is that the same distributions and packaging software cannot be used on other operating systems.

Design Goals of EPM

EPM was designed from the beginning to build binary software distributions using a common software specification format. The same distribution files work for all operating systems and all distribution formats. Supporting source code distributions was not a goal since most RPM and Debian source distributions are little more than wrapping around a compressed tar file containing the source files and a configure script.

Over the years, additional features have made their way into EPM to support more advanced software packages. Whenever possible, EPM emulates a feature if the vendor packager does not support it natively.

Resources

The EPM web site provides access to the current software and documentation for EPM:

`http://www.easysw.com/epm/`

The EPM source code can be downloaded in compressed tar files or via the popular CVS software. Please see the EPM web site for complete instructions.

The Easy Software Products news server provides several newsgroups for EPM. You can access it at:

`news.easysw.com`

Commercial support for EPM is available from Easy Software Products and is one way to contribute to the continued development of EPM. The other way to contribute is by donating code, examples, and bug fixes. If you have adapted EPM for another operating system or have added a new feature that you feel will be generally useful, please contribute it!

2 – Building EPM

This chapter shows how to configure, build, and install the ESP Package Manager.

Requirements

EPM requires very little pre−installed software to work. Most items will likely be provided as part of your OS. Your development system will need a C compiler, the `make(1)` program (GNU, BSD, and most vendor `make` programs should work), the Bourne (or Korn or Bash) shell (`sh(1)`), and `gzip(1)`.

The optional graphical setup program requires a C++ compiler, the FLTK library, version 1.1.x, and (for UNIX/Linux) the X11 libraries. FLTK is available at the following URL:

> `http://www.fltk.org/`

Your end−user systems will require the Bourne (or Korn or Bash) shell (`sh`), the `df(1)` program, the `tar(1)` program, and the `gzip(1)` program to install portable distributions. All but the last are standard items, and most vendors include `gzip` as well.

> **Note:**
>
> The `gzip` program is only required to uncompress the software distribution *.tar.gz* file. If you supply the uncompressed *.tar* file or its contents, then `gzip` is not required on the end−user system.

EPM can also generate vendor−specific distributions. These require the particular vendor tool, such as `rpm(8)` and `dpkg(8)`, to generate the software distribution on the development system and load the software distribution on the end−user system.

Configuring the Software

EPM uses GNU `autoconf(1)` to configure itself for your system. The `configure` script is used to configure the EPM software, as follows:

```
./configure ENTER
```

Choosing Compilers

If the `configure` script is unable to determine the name of your C or C++ compiler, set the `CC` and `CXX` environment variables to point to the C and C++ compiler programs, respectively. You can set these variables using the following commands in the Bourne, Korn, or Bash shells:

```
export CC=/foo/bar/gcc ENTER
export CXX=/foo/bar/gcc ENTER
```

If you are using C shell or TCsh, use the following commands instead:

```
setenv CC /foo/bar/gcc ENTER
setenv CXX /foo/bar/gcc ENTER
```

Run the `configure` script again to use the new commands.

Choosing Installation Directories

The default installation prefix is */usr*, which will place the EPM programs in */usr/bin*, the setup GUI in */usr/lib/epm*, and the man pages in */usr/man*. Use the `−−prefix` option to relocate these files to another directory:

```
./configure −−prefix=/usr/local ENTER
```

The `configure` script also accepts the `−−bindir`, `−−libdir`, and `−−mandir` options to relocate each directory separately, as follows:

```
./configure −−bindir=/usr/local/bin −−libdir=/usr/local/lib \
    −−mandir=/usr/local/share/man ENTER
```

Options for the Setup GUI

The setup GUI requires the FLTK library. The configure script will look for the `fltk−config` utility that comes with FLTK 1.1.x. Set the FLTKCONFIG environment variable to the full path of this utility if it cannot be found in the current path:

```
setenv FLTKCONFIG /foo/bar/bin/fltk−config ENTER
```

or:

```
FLTKCONFIG=/foo/bar/bin/fltk-config ENTER
export FLTKCONFIG
```

Building the Software

Once you have configured the software, type the following command to compile it:

```
make ENTER
```

Compilation should take a few minutes at most. Then type the following command to determine if the software compiled successfully:

```
make test ENTER
Portable distribution build test PASSED.
Native distribution build test PASSED.
```

The `test` target builds a portable and native distribution of EPM and reports if the two distributions were generated successfully.

Installing the Software

Now that you have compiled and tested the software, you can install it using the `make` command or one of the distributions that was created. You should be logged in as the super−user unless you specified installation directories for which you have write permission. The `su(8)` command is usually sufficient to install software:

```
su ENTER
```

Installing Using the `make` Command

Type the following command to install the EPM software using the `make` command:

```
make install ENTER
Installing EPM setup in /usr/lib/epm
Installing EPM programs in /usr/bin
Installing EPM manpages in /usr/man/cat1 and /usr/man/man1
Installing EPM documentation in /usr/share/doc/epm
```

Installing Using the Portable Distribution

The portable distribution can be found in a subdirectory named using the operating system, version, and architecture. For example, the subdirectory for a Linux 2.4.x system on an Intel−based system would be *linux−2.4−intel*. The subdirectory name is built from the following template:

```
os−major.minor−architecture
```

The `os` name is the common name for the operating system. Table 2.1 lists the abbreviations for most operating systems:

Table 2.1: Operating System Name Abbreviations

Operating System	Name
AIX	`aix`
Compaq Tru64 UNIX Digital UNIX OSF/1	`tru64`
Darwin	`darwin`
FreeBSD	`freebsd`
HP−UX	`hpux`
IRIX	`irix`
Linux	`linux`
MacOS X	`darwin`
NetBSD	`netbsd`
OpenBSD	`openbsd`
Solaris	`solaris`

The `major.minor` string is the operating system version number. Any patch revision information is stripped from the version number, as are leading characters before the major version number. For example, HP−UX version B.11.11 will result in a version number string of `11.11`.

The `architecture` string identifies the target processor. Table 2.2 lists the supported processors:

Table 2.2: Processor Architecture Abbreviations

Processor(s)	Abbreviation
Compaq Alpha	`alpha`
HP Precision Architecture	`hppa`
INTEL 80x86	`intel`
MIPS RISC	`mips`
IBM Power PC	`powerpc`
SPARC MicroSPARC UltraSPARC	`sparc`

Once you have determined the subdirectory containing the distribution, type the following commands to install EPM from the portable distribution:

```
cd os-major.minor-architecture ENTER
./epm.install ENTER
```

The software will be installed after answering a few yes/no questions.

Installing Using the Native Distribution

The `test` target also builds a distribution in the native operating system format, if supported. Table 2.3 lists the native formats for each supported operating system and the command to run to install the software.

Table 2.3: Native Operating System Formats

Operating System	Format	Command
AIX	`aix`	`installp -d`*`directory`* `epm`
Compaq Tru64 UNIX Digital UNIX OSF/1	`setld`	`setld -a` *`directory`*`???`
FreeBSD NetBSD OpenBSD	`bsd`	`cd` *`directory`* `pkg_add epm`
HP-UX	`depot`	`swinstall -f` *`directory`*
IRIX	`inst`	`swmgr -f` *`directory`*
Linux	`rpm`	`rpm -i` *`directory`*`/epm-3.0.rpm`
MacOS X	`osx`	Double-click on the `.pkg` folder in the finder.
Solaris	`pkg`	`pkgadd -d` *`directory`* `epm`

3 – Packaging Your Software with EPM

This chapter describes how to use EPM to package your own software packages.

The Basics

EPM reads one or more software "list" files that describe a single software package. Each list file contains one or more lines of ASCII text containing product or file information. Comments start with the # character, directives start with the % character, variable start with the $ character, and files, directories, and symlinks start with a letter.

Product Information

Every list file needs to define the product name, copyright, description, license, README file, vendor, and version:

```
%product Kung Foo Firewall
%copyright 1999-2002 by Foo Industries, All Rights Reserved.
%vendor Foo Industries
%license COPYING
%readme README
%description Kung Foo firewall software for your firewall.
%version 1.2.3p4 1020304
```

The `%license` and `%readme` directives specify files for the license agreement and README files for the package, respectively.

The %product, %copyright, %vendor, and %description directives take text directly from the line.

The %version directive specifies the version numbers of the package. The first number is the human−readable version number, while the second number is the integer version number. If you omit the integer version number, EPM will calculate one for you.

Files, Directories, and Symlinks

Each file in the distribution is listed on a line starting with a letter. The format of all lines is:

```
type mode owner group destination source options
```

Regular files use the letter f for the type field:

```
f 755 root sys /usr/bin/foo foo
```

Configuration files use the letter c for the type field:

```
c 644 root sys /etc/foo.conf foo.conf
```

Directories use the letter d for the type field and use a source path of "−":

```
d 755 root sys /var/spool/foo −
```

Finally, symbolic links use the letter l (lowercase L) for the type field:

```
l 000 root sys /usr/bin/foobar foo
```

The source field specifies the file to link to and can be a relative path.

Wildcards

Wildcard patterns can be used in the source field to include multiple files on a single line:

```
f 0444 root sys /usr/share/doc/foo *.html
```

Building a Software Distribution

The epm(1) program is used to build software distributions from list files. To build a portable software distribution for an application called "foo", type the following command:

```
epm foo ENTER
```

If your application uses a different base name than the list file, you can specify the list filename on the command−line as well:

```
epm foo bar.list ENTER
```

EPM can also produce vendor−specific distributions using the −f option:

```
epm −f format foo bar.list ENTER
```

The *format* option can be one of the following keywords:

- `aix` − AIX software distribution.
- `bsd` − FreeBSD, NetBSD, or OpenBSD software distribution.
- `depot` or `swinstall` − HP−UX software distribution.
- `dpkg` − Debian software distribution.
- `inst` or `tardist` − IRIX software distribution.
- `native` − "Native" software distribution (RPM, INST, DEPOT, PKG, etc.) for the platform.
- `osx` − MacOS X software distribution.
- `pkg` − Solaris software distribution.
- `portable` − Portable software distribution (default).
- `rpm` − RedHat software distribution.
- `setld` − Tru64 (setld) software distribution.

Everything in the software list file stays the same − you just use the `-f` option to select the format. For example, to build an RPM distribution of EPM, type:

```
epm -f rpm epm
```

The result will be an RPM distribution file instead of the portable distribution file.

Installing the Software Package

Once you have created the software distribution, you can install it. Portable distributions create an install script called *product.install*, where "product" is the name of the package:

```
cd os-release-arch ENTER
./product.install ENTER
```

After answering a few yes/no questions, the software will be installed. To bypass the questions, run the script with the `now` argument:

```
cd os-release-arch ENTER
./product.install now ENTER
```

Including the Setup GUI

EPM also provides an optional graphical setup program. To include the setup program in your distributions, create a product logo image in XPM format and use the `--setup-image` option when creating your distribution:

```
epm --setup-image foo.xpm foo ENTER
```

4 – Advanced Packaging with EPM

This chapter describes the advanced packaging features of EPM.

Including Other List Files

The `%include` directive includes another list file:

```
%include filename
```

Includes can be nested, usually up to 250 levels (depends on the host operating system and libraries.)

Conflicts, Provides, Replaces, and Requires

Software conflicts and requirements are specified using the `%incompat` and `%requires` directives. If your software replaces another package, you can specify that using the `%replaces` directive (`%replaces` is silently mapped to `%conflicts` when the distribution format does not support package replacement.) If your package provides certain functionality associated with a standard name, the `%provides` directive can be used.

Dependencies are specified using the package name and optionally the lower and upper version numbers:

```
%requires foobar
%requires foobar 1.0
%incompat foobar
%incompat foobar 0.9
```

```
%replaces foobar
%replaces foobar 1.2 3.4
%provides foobar
```

or the filename:

```
%requires /usr/lib/libfoobar.so
%incompat /usr/lib/libfoobar.so.1.2
```

Package dependencies are currently enforced only for the same package format, so a portable distribution that requires package "foobar" will only look for an installed "foobar" package in portable format.

Filename dependencies are only supported by the Debian, portable, and RPM distribution formats.

Scripts

Bourne shell script commands can be executed before or after installation, patching, or removal of the software. The `%preinstall` and `%postinstall` directives specify commands to be run before and after installation, respectively:

```
%preinstall echo Command before installing
%postinstall echo Command after installing
```

Similarly, the `%prepatch` and `%postpatch` directives specify commands to be executed before and after patching the software:

```
%prepatch echo Command before patching
%postpatch echo Command after patching
```

Finally, the `%preremove` and `%postremove` directives specify commands that are run before and after removal of the software:

```
%preremove echo Command before removing
%postremove echo Command after removing
```

To include an external script file, use the `<filename` notation:

```
%postinstall <filename
```

To include multiple lines directly, use the `<<string` notation:

```
%postinstall <<EOF
echo Command before installing
/usr/bin/foo
EOF
```

Note that all commands specified in the list file will use the variable expansion provided by EPM, so be sure to quote any dollar sign (`$`) characters in your commands. For example, "$foo" is replaced by the value of "foo", but "$$foo" becomes "$foo".

Conditional Directives

The `%system` directive can match or not match specific operating system names or versions. The operating system name is the name reported by `uname` in lowercase, while the operating system version is the major and minor version number reported by `uname -r`:

`%system irix`
: Only include the following files when building a distribution for the IRIX operating system.

`%system linux-2.0`
: Only include the following files when building a distribution for Linux 2.0.x.

`%system !irix !linux-2.0`
: Only include the following files when building a distribution for operating systems other than IRIX and Linux 2.0.x.

The special name `all` is used to match all operating systems:

```
%system all
```

For format-specific files, the `%format` directive can be used:

`%format rpm`
: Only include the following files when building an RPM distribution.

`%format !rpm`
: Only include the following files when not building an RPM distribution.x.

`%format all`
: Include the following files for all types of distributions.

Finally, EPM can conditionally include lines using the `%if`, `%elseif`, `%ifdef`, `%elseifdef`, `%else`, and `%endif` directives. `%if` directives include the text that follows if the named variable(s) are defined to a non-empty string, while `%ifdef` directives only include the text if the named variable(s) are defined to any value.

Protecting Object Files from Stripping

The `nostrip()` option can be included at the end of a file line to prevent EPM from stripping the symbols and debugging information from the file:

```
f 755 root sys /usr/lib/libfoo.so libfoo.so nostrip()
```

Software Patches

EPM supports portable software patch distributions which contain only the differences between the original and patch release. Patch files are specified using uppercase letters for the affected files. In the following example, the files */usr/bin/bar* and */etc/foo.conf* are marked as changed since the original release:

```
f 755 root sys /usr/bin/foo foo
F 755 root sys /usr/bin/bar bar
f 755 root sys /usr/share/man/man1/foo.1 foo.man
f 755 root sys /usr/share/man/man1/bar.1 bar.man
C 644 root sys /etc/foo.conf foo.conf
```

Variables

EPM imports the current environment variables for use in your list file. You can also define new variable in the list file or on the command−line when running EPM.

Variables are defined by starting the line with the dollar sign (`$`) followed by the name and value:

```
$name=value
$prefix=/usr
$exec_prefix=${prefix}
$bindir=$exec_prefix/bin
```

Variable substitution is performed when the variable is defined, so be careful with the ordering of your variable definitions.

Also, any variables you specify in your list file will be overridden by variables defined on the command−line or in your environment, just like with `make`. This can be a useful feature or a curse, depending on your choice of variable names.

As you can see, variables are referenced using the dollar sign (`$`). As with most shells, variable names can be surrounded by curly braces (`${variable}`) to explicitly delimit the name.

If you need to insert a `$` in a filename or a script, use `$$`:

```
%install echo Enter your name:
%install read $$name
%install echo Your name is $$name.
```

Init Scripts

Initialization scripts are generally portable between platforms, however the location of initialization scripts varies greatly.

The `i` file type can be used to specify and init script that is to be installed on the system. EPM will then determine the appropriate init file directories to use and create any required symbolic links to support the init script:

```
i 755 root sys foo foo.sh
```

The previous example creates an init script named *foo* on the end−user system and will create symbolic links to run levels 0, 2, 3, and 5 as needed, using a sequence number of 00 (or 000) for the shutdown script and 99 (or 999) for the startup script.

To specify run levels and sequence numbers, use the `runlevel()`, `start()`, and `stop()` options:

```
i 755 root sys foo foo.sh "runlevel(02) start(50) stop(30)"
```

A − GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

> a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
>
> b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
>
> c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

> a. Accompany it with the complete corresponding machine−readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine−readable

copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty−free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made

generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

B − Command Reference

epm

Create software packages

Synopsis

epm [−a *architecture*] [−f *format*] [−g] [−k] [−n[mrs]] [−s *setup.xpm*] [−−output−dir *directory*] [−−setup−image *setup.xpm*] [−−setup−program */foo/bar/setup*] [−−setup−types *setup.types*] [−v] [*name=value name=value*] product [*listfile*]

Description

epm generates software packages complete with installation, removal, and (if necessary) patch scripts. Unless otherwise specified, the files required for *product* are read from a file named "*product*.list".

The *−a* option ("architecture") specifies the actual architecture for the software. Without this option the generic processor architecture is used ("intel", "sparc", "mips", etc.)

The *−f* option ("format") specifies the distribution format:

aix
: Generate an AIX distribution suitable for installation on an AIX system.

bsd
: Generate a BSD distribution suitable for installation on a FreeBSD, NetBSD, or OpenBSD system.

deb
: Generate a Debian distribution suitable for installation on a Debian Linux system.

inst, tardist
: Generate an IRIX distribution suitable for installation on an system running IRIX.

native
: Generate an native distribution. This uses *rpm* for Linux, *inst* for IRIX, *pkg* for Solaris, *swinstall* for HP−UX, *bsd* for FreeBSD, NetBSD, and OpenBSD, and *osx* for MacOS X. All other operating systems default to the *portable* format.

osx
: Generate a MacOS X software package.

pkg
: Generate an ATTsoftware package. These are used primarily under Solaris.

portable
: Generate a portable distribution based on shell scripts and tar files. The resulting distribution is installed and removed the same way on all operating systems. [default]

rpm
: Generate a Red Hat Package Manager ("RPM") distribution suitable for installation on a Red Hat Linux system.

setld
: Generate a Tru64 (setld) software distribution.

slackware
: Generate a Slackware Linux software distribution.

swinstall, depot
: Generate a HP−UX software distribution.

Executable files in the distribution are normally stripped of debugging information when packaged. To disable this functionality use the *−g* option.

Intermediate (spec, etc.) files used to create the distribution are normally removed after the distribution is created. The *−k* option keeps these files in the distribution directory.

The *−s* and *−−setup−image* options ("setup") include the ESP Software Wizard with the specified XPM image file with the distribution. This option is currently only supported by portable distributions.

The *−−setup−program* option specifies the setup executable to use with the distribution. This option is currently only supported by portable distributions.

The *−−setup−types* option specifies the *setup.types* file to include with the distribution. This option is currently only supported by portable distributions.

The *−−output−dir* option specifies the directory to place output file into. The default directory is based on the operating system, version, and architecture.

The *−v* option ("verbose") increases the amount of information that is reported. Use multiple v's for more verbose output.

Distributions normally are named "product−version−system−release−machine.ext" and "product−version−system−release−machine−patch.ext" (for patch distributions.) The "system−release−machine" information can be customized or eliminated using the *−n* option with the appropriate trailing letters. Using *−n* by itself will remove the "system−release−machine" string from the filename entirely.

Debian, IRIX, portable, and Red Hat distributions use the extensions ".deb", ".tardist", "tar.gz", and ".rpm" respectively.

List Variables

EPM maintains a list of variables and their values which can be used to substitute values in the list file. These variables are imported from the current environment and taken from the command−line and list file as provided. Substitutions occur when the variable name is referenced with the dollar sign ($):

```
%install echo What is your name:
%install read $$name
%install echo Your name is $$name

f 0555 root sys ${bindir}/foo foo
f 0555 root sys $datadir/foo/foo.dat foo.dat
```

Variable names can be surrounded by curley brackets (${name}) or alone ($name); without brackets the name is terminated by the first slash (/), dash (−), or whitespace. The dollar sign can be inserted using $$.

Known Bugs

EPM does not currently support generation of IRIX software patches.

See Also

epm(1) − create software packages.

epminstall(1) − add a directory, file, or symlink to a list file.
mkepmlist(1) − make an epm list file from a directory.
epm.list(5) − epm list file format.

setup(1) − graphical setup program for the esp package manager.

Copyright

epminstall

Add a directory, file, or symlink to a list file.

Synopsis

epminstall *options* file1 file2 ... fileN directory

epminstall *options* file1 file2

epminstall *options* −d directory1 directory2 ... directoryN

Description

epminstall adds or replaces a directory, file, or symlink in a list file. The default list file is *epm.list* and can be overridden using the `EPMLIST` environment variable or the `−−list−file` option.

Entries are either added to the end of the list file or replaced in−line. Comments, directives, and variable declarations in the list file are preserved.

Options

epminstall recognizes the standard Berkeley *install* command options:

−b
: Make a backup of existing files (ignored, default for EPM.)

−c
: BSD old compatibility mode (ignored.)

−g group
: Set the group owner of the file or directory to *group*. The default group is "sys".

−m mode
: Set the permissions of the file or directory to *mode*. The default permissions are 0755 for directories and executable files and 0644 for non−executable files.

−o owner
: Set the owner of the file or directory to *owner*. The default owner is "root".

−s
: Strip the files (ignored, default for EPM.)

−−list−file filename.list
: Specify the list file to update.

See Also

epm(1) − create software packages.
mkepmlist(1) − make an epm list file from a directory.
epm.list(5) − epm list file format.

Copyright

mkepmlist

Make an EPM list file from a directory.

Synopsis

mkepmlist [−g *group*] [−u *user*] [−−prefix *directory*] *directory* [... *directory*]

Description

mkepmlist recursively generates file list entries for files, links, and directories. The file list is send to the standard output.

The −*g* option overrides the group ownership of the files in the specified directories with the specified group name.

The −*u* option overrides the user ownership of the files in the specified directories with the specified user name.

The −−*prefix* option adds the specified directory to the destination path. For example, if you installed files to */opt/foo* and wanted to build a distribution that installed the files in */usr/local*, the following command would generate a file list that is installed in */usr/local*:

```
mkepmlist --prefix=/usr/local /opt/foo >foo.list ENTER
```

See Also

epm(1) − create software packages.
epminstall(1) − add a directory, file, or symlink to a list file.
epm.list(5) − epm list file format.

Copyright

setup

Graphical setup program for the ESP package manager.

Synopsis

setup [*directory*]

Description

setup provides a graphical installation interface for EPM−generated portable installation packages. It presents a step−by−step dialog for collecting a list of packages to install and accepting any license agreements for those packages.

setup searches for products in the current directory or the directory specified on the command−line.

Installation Types

The default type of installation is "custom". That is, users will be able to select from the list of products and install them.

setup also supports other types of installations. The *setup.types* file, if present, defines the other installation types.

See Also

epm(1) − create software packages.
setup.types(5) − epm gui setup types file format.

Copyright

C − List File Reference

This appendix provides a complete reference for the EPM list file and setup types formats.

The EPM List File Format

Each *EPM* product has an associated list file that describes the files to include with the product. Comment lines begin with the "#" character and are ignored. All other non−blank lines must begin with a letter, dollar sign ("$"), or the percent sign ("%").

List File Directives

The following list describes all of the list file directives supported by *EPM*:

$name=value
: Sets the named variable to *value*. **Note:** Variables set in the list file are overridden by variables specified on the command−line or in the current environment.

%copyright copyright notice
: Sets the copyright notice for the file.

%description description text
: Adds a line of descriptive text to the distribution. Multiple lines are supported.

%format format [... format]
: Uses following files and directives only if the distribution format is the same as *format*.

%format !format [... format]
: Uses following files and directives only if the distribution format is not the same as *format*.

%if variable [... variable]
%if !variable [... variable]
%ifdef variable [... variable]
%ifdef !variable [... variable]
%elseif variable [... variable]
%elseif !variable [... variable]
%elseifdef variable [... variable]
%elseifdef !variable [... variable]
%else
%endif

Conditionally includes lines in the list file. The *%if* lines include the lines that follow if the named variables are (not) defined with a value. The *%ifdef* lines include the lines that follow if the named variables are (not) defined with any value. These conditional lines cannot be nested.

%include filename

Includes files listed in *filename*.

%incompat product
%incompat filename

Indicates that this product is incompatible with the named product or file.

%install script or program

Specifies a script or program to be run after all files are installed. (This has been obsoleted by the %postinstall directive)

%license license file

Specifies the file to display as the software license.

%packager name of packager

Specifies the name of the packager.

%patch script or program

Specifies a script or program to be run after all files are patched. (This has been obsoleted by the %postpatch directive)

%postinstall script or program
%postinstall <scriptfile
%postinstall <<string

Specifies a script or program to be run after all files are installed.

%postpatch script or program
%postpatch <scriptfile
%postpatch <<string

Specifies a script or program to be run after all files are patched.

%postremove script or program
%postremove <scriptfile
%postremove <<string

Specifies a script or program to be run after removing files.

%preinstall script or program
%preinstall <scriptfile
%preinstall <<string

Specifies a script or program to be run before all files are installed.

%prepatch script or program
%prepatch <scriptfile
%prepatch <<string

Specifies a script or program to be run before all files are patched.

%preremove script or program
%preremove <scriptfile

%preremove <<string
: Specifies a script or program to be run before removing files.

%product product name
: Specifies the product name.

%readme readme file
: Specifies a README file to be included in the distribution.

%remove script or program
: Specifies a script or program to be run before removing files. (This has been obsoleted by the %preremove directive)

%release number
: Specifies the release or build number of a product (defaults to 0).

%replaces product
: Indicates that this product replaces the named product.

%requires product
%requires filename
: Indicates that this product requires the named product or file.

%vendor vendor or author name
: Specifies the vendor or author of the product.

%version version number
: Specifies the version number of the product.

%system system[−release] [... system[−release]]
: Specifies that the following files should only be used for the specified operating systems and releases.

%system !system[−release] [... system[−release]]
: Specifies that the following files should not be used for the specified operating systems and releases.

%system all
: Specifies that the following files are applicable to all operating systems.

c mode user group destination source
C mode user group destination source
: Specifies a configuration file for installation. The second form specifies that the file has changed or is new and should be included as part of a patch. Configuration files are installed as "destination.N" if the destination already exists.

d mode user group destination −
D mode user group destination −
: Specifies a directory should be created when installing the software. The second form specifies that the directory is new and should be included as part of a patch.

f mode user group destination source [nostrip()]
F mode user group destination source [nostrip()]
: Specifies a file for installation. The second form specifies that the file has changed or is new and should be included as part of a patch. If the "nostrip()" option is included, the file will not be stripped before the installation is created.

f mode user group destination source/pattern [nostrip()]
F mode user group destination source/pattern [nostrip()]
: Specifies one or more files for installation using shell wildcard patterns. The second form specifies that the files have changed or are new and should be included as part of a patch. If the "nostrip()" option is included, the file will not be stripped before the installation is created.

i mode user group service−name source ["options"]
I mode user group service−name source ["options"]

Specifies an initialization script for installation. The second form specifies that the file has changed or is new and should be included as part of a patch. Initialization scripts are stored in */etc/software/init.d* and are linked to the appropriate system−specific directories for run levels 0, 2, 3, and 5. Initialization scripts **must** accept at least the *start* and *stop* commands. The optional *options* following the source filename can be any of the following:

order(string)
: Specifies the relative startup order compared to the required and used system functions. Supported values include First, Early, None, Late, and Last (OSX only).

provides(name(s))
: Specifies names of system functions that are provided by this startup item (OSX only).

requires(name(s))
: Specifies names of system functions that are required by this startup item (OSX only).

runlevels(levels)
: Specifies the run levels to use.

start(number)
: Specifies the starting sequence number from 00 to 99.

stop(number)
: Specifies the ending sequence number from 00 to 99.

uses(name(s))
: Specifies names of system functions that are used by this startup item (OSX only).

l mode user group destination source

L mode user group destination source
: Specifies a symbolic link in the installation. The second form specifies that the link has changed or is new and should be included as part of a patch.

R mode user group destination
: Specifies that the file is to be removed upon patching. The *user* and *group* fields are ignored. The *mode* field is only used to determine if a check should be made for a previous version of the file.

List Variables

EPM maintains a list of variables and their values which can be used to substitute values in the list file. These variables are imported from the current environment and taken from the command−line and list file as provided. Substitutions occur when the variable name is referenced with the dollar sign ($):

```
%postinstall <<EOF
echo What is your name:
read $$name
echo Your name is $$name
EOF

f 0555 root sys ${bindir}/foo foo
f 0555 root sys $datadir/foo/foo.dat foo.dat
```

Variable names can be surrounded by curley brackets (${name}) or alone ($name); without brackets the name is terminated by the first slash (/), dash (−), or whitespace. The dollar sign can be inserted using $$.

The setup.types File

The EPM **setup** program normally presents the user with a list of software products to install, which is called a "custom" software installation.

If a file called *setup.types* is present in the package directory, the user will instead be presented with a list of instal− lation types. Each type has an associated product list which determines the products that are installed by default. If a type has no products associated with it, then it is treated as a custom installation and the user is presented with a list of packages to choose from.

The *setup.types* file is an ASCII text file consisting of type and product lines. Comments can be inserted by starting a line with the pound sign (#). Each installation type is defined by a line starting with the word `TYPE`. Products are defined by a line starting with the word `INSTALL`:

```
TYPE Typical End−User Configuration
INSTALL foo
INSTALL foo−help
TYPE Typical Developer Configuration
INSTALL foo
INSTALL foo−help
INSTALL foo−devel
INSTALL foo−examples
TYPE Custom Configuration
```

In the example above, three installation types are defined. Since the last type includes no products, the user will be presented with the full list of products to choose from.

D − Release Notes

This appendix lists the change log for each release of the EPM software.

Changes in EPM v3.7

- AIX output now correctly lists file sizes by directory and handles /opt properly.
- RPM output now correctly handles installing, upgrading, and removing init scripts.
- RPM output did not work with filenames that contained a dollar sign ($).
- Added Slackware packager support based upon a patch contributed by Alec Thomas.
- The file copy code reported write errors for the source filename and not the (correct) destination filename.
- Fixed the handling of absolute output directories when generating RPM packages.
- The configure script did not support the OPTIM environment variable for custom optimization settings.
- Portable packages updated the permissions of configuration files before they were copied.
- Portable installations did not remove empty installation directories when the remove script was run.
- Portable patch installation did not correctly determine when root or /usr files were present in the patch.
- OSX packages incorrectly looked in /System/Library for the init scripts instead of /Library.
- RPM building did not work properly when −−output−dir was specified using an absolute path.
- Made cosmetic changes to the setup/uninstall GUIs.
- The setup GUI did not support software patches.
- The documentation incorrectly specified runlevels() instead of runlevel().
- The portable distributions incorrectly used /usr/local/src/rc.d as a fallback location for init scripts.

Changes in EPM v3.6

- Added a GUI uninstall program to be distributed with portable distributions with a setup image.
- MacOS X portable packages now support graphical setup and uninstall using the Apple authorization API.
- Debian packages did not include the (required) trailing period when running the update−rc.d script.
- BSD packages now create directories using postinstall commands instead of listing them directly. This should eliminate errors from the FreeBSD pkg_delete command.
- File dependencies were incorrectly specified in RPM spec files.
- IRIX portable distributions didn't write the chkconfig commands properly.
- The mkepmlist utility didn't support files as well as directories.
- EPM didn't do variable expansion of imported files ("<foo.txt") or in−line data ("<<FOO")
- Now build gzip'd depot files as well as the tar.gz files when creating HP−UX software packages.
- Now correctly use −−libdir setting to locate the setup GUI.
- Now use rpmbuild command, if available, to build RPM files since newer versions of RPM may not map the "−bb" option to build a package.
- Now set the RPMDIR environment variable when building with older versions of RPM that don't understand the "topdir_" variable.
- Now handle dependencies of the form "package >= version" in Debian packages.
- Portable distributions that didn't have any files in /usr or in / would look for a non−existent .sw or .ss file.
- EPM's sample list file didn't include the man pages for setup or setup.types, and installed the epm list file format man page in the wrong directories.
- EPM would quote the ":" character in filenames but didn't need to.
- EPM tried to move the wrong RPM file on non−intel systems.

Changes in EPM v3.5.1

- OSX packages did not set the "install as root" package type, so package installation usually failed.
- OSX packages installed init scripts in /System/Library/StartupItems, but non−Apple packages should be installed in /Library/StartupItems.
- Added support for "requires(foo)", "uses(foo)", "provides(foo)", and "order(foo)" as options for init scripts. These options are currently only used when creating OSX packages.

Changes in EPM v3.5

- Added support for MacOS X package generation.
- No longer need/use RPMDIR when building RPMs, just set the "topdir_" variable in the spec file.
- The portable removal scripts didn't correctly write the list of init scripts to remove.
- Added a new −−output−dir option contributed by Geoffrey Wossum.
- Not all implementations of the "id" command support the "−u" option, so EPM now looks at the default output.
- RPM dependencies with a single version number didn't get written to the spec file properly.
- Added support for file options − nostrip(), runlevel(12345), start(nn), and stop(nn).
- Filenames can now contain spaces, either by putting the full name in quotes ("file with spaces") or using the backslash character (file\ with\ spaces).
- The mkepmlist utility didn't handle symlinks properly.
- BSD packages needed to list the directories to remove separately and in reverse order.

Changes in EPM v3.4

- No longer install init scripts in run levels 2 and 5 under Solaris, which runs all init scripts in each run level.
- The tar files produced by EPM didn't conform to the POSIX 1003.1 spec. EPM now puts the POSIX version number (00) and supports "long" filenames up to 255 characters in length.
- AIX packages did not use the correct path for files placed in the root partition.
- Now install init scripts for *BSD in /usr/local/etc/rc.d.
- Portable installation scripts now issue chown and chgrp commands for all installed files that are not owned by user root.
- No longer use or pad empty tar files, which saves 5k of disk space per distribution.

Changes in EPM v3.3

- UnixWare 7 needs the absolute path when transferring an AT&T package directory to a .pkg file.
- Now use the "id" command (POSIX) instead of "whoami" (BSD) to check that the installing user is root.
- Various fixes for OpenServer.
- Now use the "−ln" option when checking for the size of the distribution files in portable distributions.
- New C implementation of mkepmlist, based on a contribution from Andreas Voegele.
- The portable install and patch scripts now change the permissions of the installed "remove" script to 544.
- The wildcard character * did not match 0 characters if the pattern and the filename string differed only by the trailing * (e.g. "Courier" and "Courier*").
- IRIX pre/postremove scripts are now copied after installation so that they can be executed when removing the inst/tardist package.
- HP−UX postinstall/preremove scripts would execute init scripts from /sbin/init.d/sbin/init.d.
- The copyright string wasn't being quoted in portable installation scripts.
- EPM now checks to see if an executable file is a shell script before running the "strip" command.

Changes in EPM v3.2.1

- The check for Darwin (MacOS X) in the portable installation scripts was using "==" instead of just "=".

Changes in EPM v3.2

- Added "SHELL=/bin/sh" line to portable installation, patch, and remove scripts in case the root shell is not /bin/sh or a compatible shell.
- The epminstall utility didn't support the EPMLIST environment variable as documented.
- The "native" distribution format is now "deb" if the "dpkg" command is installed in /usr/bin.
- Debian packages did not use the release number in the Version: keyword.
- Changed the portable installation script support for init scripts to look for scripts in both rc2.d and rc3.d, and to check for the existence of all rc directories before installing into them.
- Changed the portable installation script to prepend /bin, /usr/bin, and /usr/ucb to the beginning of the PATH variable so the "tar", "rm", and "mv" commands can be found in whatever directory the local system uses.
- AIX packaging now works.
- Added support for the Darwin (MacOS X) tar command in portable packages (sudo ./foo.install :)

- IRIX inst packages incorrectly had the postinstall commands in the preinstall and removal scripts.
- The setup GUI now requires FLTK 1.1.x.

Changes in EPM v3.1

- Added support for "<<end ... end" and "<filename" to insert descriptions and commands in−line and from a file.
- Added new −−software−dir option which specifies the location of the EPM software directory (default /etc/software).
- Added new %if, %ifdef, %elseif, %elseifdef, %else, and %endif directives for list files (addition contributed by J. Nordell.)
- The GUI setup program left the "Next" button enabled after the license check.
- Fixed the dependency strings created for HP−UX swinstall (fix contributed by R. Begg.)
- Wasn't installing the man pages in section 5.
- When generating Debian packages, the DEBIAN directory might not have the correct permissions due to a restrictive umask. EPM now forces the correct permissions for the package archive.
- HP−UX swinstall packages now use the prerequisites rule instead of corequisites to ensure that the %requires dependency is enforced.
- Dependencies for Debian packages are now listed on a single line per type rather than one line per dependency.
- EPM now detects RPM 4.0.3, which (mysteriously) now uses "−−target arch" instead of "−−target=arch".

Changes in EPM v3.0

- Added new documentation.
- Added new "−a arch" option to support specific architectures (e.g. i586, i686, ultrasparc, etc.)
- Added support for the *BSD package format using pkg_create.
- Added support for the AIX package format using the backup program.
- Added new epminstall utility to build list files from "make install" targets.
- Added snprintf/vsnprintf functions for systems that don't provide them, and use snprintf and vsnprintf for all formatted strings that aren't just numbers.
- Added new run_command() function to replace use of system() function.
- Added new %provides directive.
- Revamped the setup GUI, including support for installation types in the setup GUI, so that you can select groups of products or choose a custom installation.
- The mkepmlist utility had a bad regular expression that thought that any argument (directory names, etc.) with a dash ("−") in it was an unsupported option.
- Fixed a bug in the removal script: config files were removed by the remove script...
- Changed the config file install logic to copy the new config file (instead of moving it), so that an unchanged config file can be detected and removed. This provides the best of both worlds: unchanged config files are update automatically by an upgrade/install, while changed ones are preserved.

Changes in EPM v2.8

- The line breaking code did not include an extra space, so filenames in the portable install/patch/remove scripts would be joined instead of separated.
- The setup GUI did not allow the user to toggle a software product for distribution if the product was selected (nav box around it).

- The mkepmlist program didn't get the permissions of each file (just the parent directory.)
- The portable installation scripts used the −L or −h option to test for symlinks with the test command. The choice of option was based on the build platform, making the script non−portable. Now use −h exclusively since it is supported on all UNIX's we have access to, even with GNU test even though it isn't documented...

Changes in EPM v2.7

- Fixed a bug in the configuration script with the −−with−fltk−includes option.
- Tru64 UNIX distributions now use the name "tru64" instead of "dunix". "dunix" is still supported in list files for compatibility with old list files.
- Added support for portable scripts under AIX.
- Fixed the space checking code in portable installation scripts.
- Now break up long lines in the portable install/patch/ remove scripts.

Changes in EPM v2.6

- Changed the automatic version number generation code to properly handle patch, beta, and pre releases.
- Added support for release numbers in RPM files.
- Added support for version number ranges in dependencies, either as "low−version high−version", "< version", or "> version".
- Eliminated some GCC warnings about using a char to index into an array.
- Added a disk space check to the portable installation scripts.
- Added a new mkepmlist utility, based on a Perl script by Christian Lademann.
- Added a "keep files" option (−k) to epm to keep the intermediate (spec, etc.) files around after building the binary distribution.
- Added support for Tru64 UNIX software packages (setld).
- Patch distributions were incorrectly backing up the original files, causing the original backup to be lost.
- Pre/post install/remove scripts were not using the right filename for Solaris PKG distributions.

Changes in EPM v2.5

- Added support for pre−install, post−install, pre−patch, post−patch, pre−remove, and post−remove commands.
- There was no way to use a literal $ in scripts or in filenames. Use $$ to include a single $.
- The config and license file support for AT&T software packages did not check to see if the source file had an absolute path. This would produce an invalid prototype file.
- The RPM −−target option was not being called with an equal sign, which caused problems with RPM 4.0.
- Updated the Debian packager to use the prerm and postrm script names to match reality.
- Updated the Debian packager to support the Replaces dependency.
- Updated the portable and RPM distributions to check for the new SuSE 7.1 init.d directories.
- RPM distributions now use %config(noreplace) for config files, to duplicate the behavior that is expected.
- The portable scripts now use the autoconf echo test to determine the proper options for echo (−n or \c), rather then hardcoding this based on the build system.

Changes in EPM v2.4

- The [] wildcard matching did not skip over the character that was matched. This prevented matches in most cases...

Changes in EPM v2.3

- Fix for an incredibly stupid bug in the portable distribution code − was using ! instead of ~ to mask off the write permission bits in the distribution archive.
- Now use getpwuid() instead of getlogin() to get the username of the packager.
- The RPM distributions now use the same init.d script logic as portable distributions. This should make them portable to all known Linux distributions as well as avoid a *very* nasty installer bug in RedHat 7.0.
- The HP−UX swinstall code did not properly handle directories or config files.
- The [] wildcard matching rule did not accept ranges (e.g. "[a−z]", "[0−9]", etc.)
- Added VPATH support and distribution targets to Makefile.
- Added support for defining variables in list files; the format is "$name=value".
- The variable expansion code didn't check for ${name}.

Changes in EPM v2.2

- New HTML documentation files.
- Updated the BuildRoot directive in RPM spec files to be an absolute path; RedHat 6.2's version of RPM adds a leading slash otherwise.
- IRIX defaults to run level 2...
- The setup GUI now displays an error message if run by a non−root user.
- The setup GUI now provides "Install All" and "Install None" buttons in the software selection pane.
- Added a "native" distribution format to select the native format for a particular OS (Linux defaults to RPM format...)
- The tar file generation code now always appends at least 2 zeroed blocks to the end of the archive. This eliminates error messages from Solaris tar and seems to be compatible with all other tar programs.
- Added the SuSE RPM directory to the standard search path.
- Added support for a new %packager directive.
- The strip command used was redirecting stderr before redirecting stdout.
- The portable distributions now set the umask to avoid problems with buggy tar programs and Linux distributions.
- Added command−line option to specify the location of the setup program.
- Added support for wildcards in source filenames.
- The OS version number is now truncated to only contain the major and minor release numbers.

Changes in EPM v2.1

- Moved setup program to /usr/lib/epm ($prefix/lib/epm) to avoid name clash with RedHat setup program.
- Added Debian distribution files from Jeff Licquia.
- Configure script changes for GCC 2.95.x and Solaris.
- Portability fixes.

- Now look for RPMS in different "standard" locations after building them; the RPMDIR environment variable can be used to override the default locations.
- The sample project list file (epm.list) was missing from the 2.0 distribution.
- Now check for write permission in /usr by writing a test file (/usr/.writetest); this should make diskless client installations more reliable.
- Added support for variables on the command line (name=value); insert into project filenames using $name.
- Variable expansion is now done on all lines and fields. This allows variables to be used in scripts and in the permissions field, for example.
- Now only specify run levels 0 and 3 for init scripts (0, 3, and 5 for Linux.)
- Now support init scripts in /sbin/init.d and /sbin/init.d/rcN.d (SuSE.)
- RPM distributions should now work OK for non−Red Hat based systems, in particular for init scripts.
- PKG distributions are now also generated in the "package stream" format as well as the directory and tar.gz file formats.

Changes in EPM v2.0

- New "−f" option to generate vendor−specific software distributions. Now support AT&T, Debian, HP−UX, IRIX, and Red Hat software distributions.
- New "−s" option to include the ESP Software Wizard (GUI) with portable distributions.
- The "−t" option (test) is no longer supported.
- New "−v" option to control the amount of information that is reported.
- New graphical setup program for portable distributions.
- New "description" directive.
- New "format" directive.
- New "include" directive.
- New "replaces" directive.
- Portable distributions should now be more portable.

Changes in EPM v1.7

- The %requires and %incompat directives now support specification of files as well as products.
- The init script installation code now creates a link in the init.d subdirectory to avoid frustrating well−trained fingers.
- The progress messages for shared and non−shared software were the same.

Changes in EPM v1.6

- Installation archives were missing the ".ss" and ".pss" files that were added to support diskless installations.
- The scripts didn't handle removing distributions that had no non−shared components.
- The scripts didn't return a non−zero exit status if the user did not agree with the license or want to install.

Changes in EPM v1.5

- Now support diskless installations; all files destined for /usr are put in a separate archive and are installed (or removed) only if /usr is read+write.

Changes in EPM v1.4

- Now map group "sys" to "system" for Digital UNIX and "root" for Linux.
- The initialization script installation now checks for the presence of run levels 4 and 5.

Changes in EPM v1.3

- Now use the "p" option to tar to ensure that file permissions are created properly. This is normally the default for the super−user, but not under Digital UNIX!
- Initialization scripts are now linked to run levels 0, 2, 3, 4, and 5.

Changes in EPM v1.2

- Patch distributions were not correctly named.
- Added new "initialization script" file types "i" and "I". The new file types place the scripts in /etc/software/init.d and make links to the appropriate system−specific rc.d directories and run the scripts to start and stop things accordingly.

Changes in EPM v1.1

- The "whoami" command isn't always in the user's path, so scripts now use a hard−coded path (setup by the configure script) to the program.
- Added a check for IRIX64 (64−bit kernel instead of n32.)
- The %system directive now supports release numbers, e.g. "irix−6.5".
- The %system directive now supports "!" (not) operator so you can do things like "%system irix !irix−6.5" to select any IRIX release except IRIX 6.5.
- Files that already exist on the system are renamed to "filename.O" on installation and back to "filename" when removed (except for config files, which don't overwrite and aren't removed.)
- Prerequisites (%required directive) now look for required product in the current directory and install it automatically if it is available and not already installed.
- The copyright notice in the installation script was not displayed if the user used the "now" option.