Software Packaging 3

Introduction

Under SVR4, the method of structuring, installing, updating, and removing software has been standardized. There are several utility programs provided which operate on groups of product and information files; these files are bundled into units called *packages*. To utilize these tools, certain conventions need to be followed. This results in a consistent methodology for the customer who is dealing with multiple packages from several vendors. It also simplifies the job of the vendor who may have multiple versions of a product or a version for several architectures.

History

An ideal software administration methodology reduces the time required to tailor a software product to a specific hardware platform, market, or user. It also minimizes the time necessary to update the product with enhanced features, or to provide corrections to features that fail to function adequately. In addition, good software administration permits changes to a product applied to a particular system to be tracked or backed out if there are dependencies on other products which are not satisfied. Finally, it should be possible to generate reports showing the entire history of that software on a given machine.

In mainframe operating systems, such as IBM's MVS, this type of software maintenance has been possible for at least the last 15 years. IBM's SMP utility is extremely sophisticated and is perhaps the largest single executable file on the entire system. Not only does it consume a great amount of system resources to run, but the disk space requirements of its database are also vast.

In the past there has not been a comprehensive set of utilities provided with UNIX to support this kind of functionality. Vendors would choose their own conventions using various combinations of utilities such as `sccs`, `make`, `grep`, `awk`, and `sed`, as well as their own directory structures, or would leave it up to the person installing the software. Most often, `/usr` or `/home` was the starting point.

Overview

In SVR4, a method was adopted for packaging software that has become part of the ABI. It essentially involves supplying certain information files in the same directory as the software itself, and then using various package utility programs to move the files to and from some distribution medium such as tape or floppy disk. There are only two information files required: one to describe the software product as a whole (`pkginfo`) and one to describe the file structure for installation (`prototype`).

Installation scripts may not be necessary because the package utilities take care of much of the work that used to be performed by scripts. If installation scripts are provided, they become part of the package files.

The other files are optional and are used for specifying dependencies on other packages, a copyright notice, space requirements, and backwards compatibility information.

Figure 3-9 has an illustration of the mechanisms of packaged software.

Creating a Package

This section discusses how to create an installable package.

The example used in this section creates a package for a device driver (ramdisk). The steps are the same for any package, although the directory and file contents will presumably differ.

1. **Name the package.**
   Each package requires an ID which can be at most 9 characters long. The convention used within Sun is `<companysymbol><package>`. In this example, the package is called `SUNWramd`, where `SUNW` is the company symbol and `ramd` is the package, which in this case is the ramdisk. The package will be installed into a directory of this name on the destination or *target* system. Care should be taken to ensure that the *id* is unique; otherwise, this package may overwrite another of the same name at installation time.

2. **Organize the package contents.**
   Decide what files are going to be in the package and where they should be installed on the target machine. It is recommended that you organize your files into a directory tree that is a mirror of what you want on the target machine so you will not have to reorganize the directory structure later. It is also possible to have a flat directory of contents.

3. **Create the `prototype` file.**
Run the `pkgproto` utility against the source directory to create an initial `prototype` file, and then edit the file. To do this, type:

```
# cd srcdir; find . -print | pkgproto > ../prototype
```

where `srcdir` is the directory containing the package contents.

The `prototype` file is created by either `pkgproto` or the developer, and is used to describe package objects.

The format of each entry in the file is:

```
<ftype> <class> <path> <access> <owner> <group>
```

where:

`ftype`	is a one-letter code signifying the type of file described in this entry.
`class`	is either `none` or the name of a grouping to which this file belongs.
`path`	is the pathname where the file is to be installed on the target. If you do not specify `BASEDIR` and the entry in the system file `/var/sadm/install/admin/default` for `BASEDIR` is `default` (the default), relocatable files will be installed into the root directory.
`access`	is the read, write, and execute access to the file (defaults to `644`).
`owner`	is the ID name of the user who owns the file (defaults to `root`).
`group`	is the group that can access the file (defaults to `other`).

An optional field, `part`, may be appended; it defaults to `1` if not present.

This output file is usually hand-edited by the developer after an initial `prototype` file is created by `pkgproto`. Alternatively, a `sed` script may be created to automate this.

```
d none SUNWramd 0755 owner group
d none SUNWramd/usr 0755 owner group
d none SUNWramd/usr/kernel 0755 owner group
d none SUNWramd/usr/kernel/drv 0755 owner group
f none SUNWramd/usr/kernel/drv/ramd 0644 owner group
f none SUNWramd/usr/kernel/drv/ramd.conf 0644 owner group
d none SUNWramd/usr/include 0755 owner group
d none SUNWramd/usr/include/sys 0755 owner group
f none SUNWramd/usr/include/sys/ramio.h 0644 owner group
d none SUNWramd/bin 0755 owner group
f none SUNWramd/bin/testram 0755 owner group
d none SUNWramd/src 0755 owner group
f none SUNWramd/src/testram.c 0644 owner group
f none SUNWramd/src/ramd.c 0644 owner group
f none SUNWramd/src/Makefile 0644 owner group
```

Figure 3-1 Example output from `pkgproto`

For the ramdisk, information and installation script file entries were added, and the owner and group changed from the developer to groups known to exist on any installation machine.

```
i pkginfo
i postinstall
i preremove
i copyright
d none SUNWramd 0755 root sys
d none SUNWramd/usr 0755 root sys
d none SUNWramd/usr/kernel 0755 root sys
d none SUNWramd/usr/kernel/drv 0755 root sys
f none SUNWramd/usr/kernel/drv/ramd 0644 root sys
f none SUNWramd/usr/kernel/drv/ramd.conf 0644 root sys
d none SUNWramd/usr/include 0755 root sys
d none SUNWramd/usr/include/sys 0755 root sys
f none SUNWramd/usr/include/sys/ramio.h 0644 root sys
d none SUNWramd/bin 0755 root sys
f none SUNWramd/bin/testram 0755 root sys
d none SUNWramd/src 0755 root sys
f none SUNWramd/src/testram.c 0644 root sys
f none SUNWramd/src/ramd.c 0644 root sys
f none SUNWramd/src/Makefile 0644 root sys
```

Figure 3-2 Example Final `prototype` *File*

4. **Create the `pkginfo` file.**
When you create the `pkginfo` file, note that several of the entries are mandatory; the `pkginfo` man page lists these entries. The following is a sample `pkginfo` file.

```
PKG=SUNWramd
NAME=Solaris 2.0 Example Ramdisk Driver
VERSION=1
CATEGORY=system
ARCH=sparc
VENDOR=Sun Microsystems
BASEDIR=/opt
```

Figure 3-3 Sample `pkginfo` *File*

The `BASEDIR` entry is the only optional entry; it defines where relocatable files are installed. To override the setting of `BASEDIR`, use the `-a` command line option to the `pkgadd` command, as in `pkgadd -a none`.

The PKG, VERSION, and ARCH settings together define the package *instance*. `pkgadd` will distinguish instances by appending an *instance number* to the package name.

5. **Create an on-disk version of the package using** `pkgmk`**.**
Run `pkgmk`. This creates a UFS format package in a `spooldir`, which must already exist. This is typically `/var/spool`.

If you have relative pathnames for the source files in the `prototype` file, as in the previous examples, use the `-b` option to prepend the path to the source directory.

```
# pkgmk -o -d `pwd`/.. -b `pwd`
## Building pkgmap from package prototype file.
## Processing pkginfo file.
WARNING: parameter <PSTAMP> set to "hoth920825181813"
WARNING: parameter <CLASSES> set to "none"
## Attempting to volumize 15 entries in pkgmap.
part  1 -- 388 blocks, 22 entries
/home/owner/pkg/../SUNWramd/pkgmap
/home/owner/pkg/../SUNWramd/pkginfo
/home/owner/pkg/../SUNWramd/reloc/SUNWramd/bin/testram
/home/owner/pkg/../SUNWramd/reloc/SUNWramd/src/Makefile
/home/owner/pkg/../SUNWramd/reloc/SUNWramd/src/ramd.c
/home/owner/pkg/../SUNWramd/reloc/SUNWramd/src/testram.c
/home/owner/pkg/../SUNWramd/reloc/SUNWramd/usr/include/sys/ramio.h
/home/owner/pkg/../SUNWramd/reloc/SUNWramd/usr/kernel/drv/ramd
/home/owner/pkg/../SUNWramd/reloc/SUNWramd/usr/kernel/drv/ramd.conf
/home/owner/pkg/../SUNWramd/install/copyright
```

Figure 3-4 Sample `pkgmk` *Run*

```
/home/owner/pkg/../SUNWramd/install/postinstall
/home/owner/pkg/../SUNWramd/install/preremove
## Packaging complete.
```

Figure 3-4 Sample `pkgmk` *Run (Continued)*

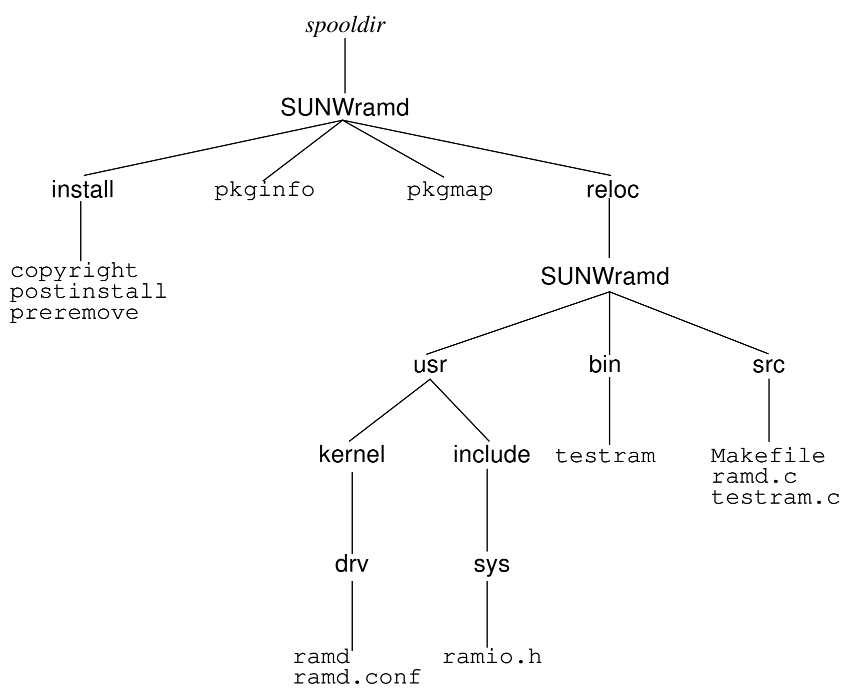

Figure 3-5 `pkgmk` *UFS Format Package Structure*

The following is the `pkginfo` file output by `pkgmk` in this example:

```
PKG=SUNWramd
NAME=Solaris 2.0 Example Ramdisk Driver
VERSION=1
CATEGORY=system
ARCH=sparc
VENDOR=Sun Microsystems
BASEDIR=/opt
PSTAMP=hoth920825181813
CLASSES=none
```

Figure 3-6 Sample `pkginfo` *Output File*

PSTAMP and CLASSES are created by pkgmk if you do not specify them. PSTAMP uniquely identifies this package from other builds of the package, the default format for it is <hostname><date/timestamp>. CLASSES is a list of the classes defined in the prototype file.

Output from pkgmk

The pkgmap file is created by pkgmk, and provides a complete listing of the package contents.

The first line indicates the number of parts, followed by the maximum part size (in 512-byte blocks). All subsequent lines are of the form:

```
part ftype class pathname mode owner group size cksum modtime
```

```
: 1 388
1 d none SUNWramd 0755 root sys
1 d none SUNWramd/bin 0755 root sys
1 f none SUNWramd/bin/testram 0755 root sys 7692 62111 714068418
1 d none SUNWramd/src 0755 root sys
1 f none SUNWramd/src/Makefile 0644 root sys 297 21372 714081106
1 f none SUNWramd/src/ramd.c 0644 root sys 22349 21963 714081103
1 f none SUNWramd/src/testram.c 0644 root sys 1431 47597 714068418
1 d none SUNWramd/usr 0755 root sys
1 d none SUNWramd/usr/include 0755 root sys
1 d none SUNWramd/usr/include/sys 0755 root sys
1 f none SUNWramd/usr/include/sys/ramio.h 0644 root sys 83 5521 714068418
1 d none SUNWramd/usr/kernel 0755 root sys
1 d none SUNWramd/usr/kernel/drv 0755 root sys
1 f none SUNWramd/usr/kernel/drv/ramd 0644 root sys 18128 24339 714068418
1 f none SUNWramd/usr/kernel/drv/ramd.conf 0644 root sys \- 49 4211
714068419
1 i copyright 34 3037 714079252
1 i pkginfo 164 13148 714766693
1 i postinstall 1608 3692 714766677
1 i preremove 714 58705 714763002
```

Figure 3-7 Sample pkgmap *File*

6. **Create a distributable package on floppy or tape using** pkgtrans**.**
   Run pkgtrans by typing:

```
# pkgtrans -s `pwd`/spooldir /dev/diskette
```

This translates the package from the UFS version to a datastream version (-s) on a diskette. The next example places the SUNWramd package on a tape drive:

```
# pkgtrans -s `pwd` /dev/rmt/0h

The following packages are available:
  1  SUNWramd     Solaris 2.0 Example Ramdisk Driver
                  (sparc) 1

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1

Insert a tape into Tape Drive.
Type [go] when ready,
  or [q] to quit: go
Transferring <SUNWramd> package instance
```

Figure 3-8 Sample Run of pkgtrans

Figure 3-9 contains a diagram of the software package file structure.

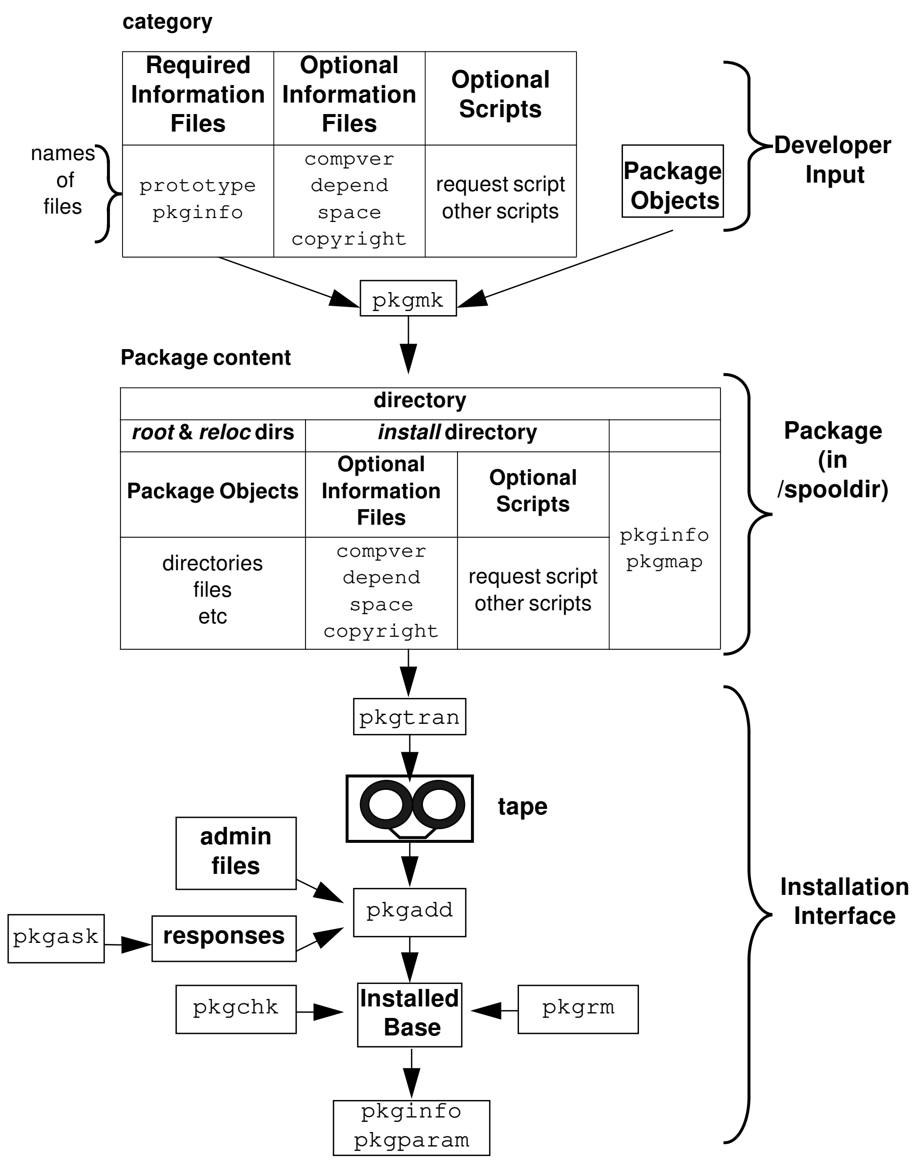

Figure 3-9 Package File Structure

Installing a Package

To install a package, use the `pkgadd` utility. To run `pkgadd`, type:

```
# pkgadd /dev/diskette [<pkgname>]
```

where <`pkgname`> is the name of your package. It may also be *all* which will cause `pkgadd` not to prompt for a package name but instead to install all of the packages on the installation media.

Note – You must be root to install or remove a package.

The following is a sample run of `pkgadd` for installation from a tape.

```
# pkgadd -d /dev/rmt/0h
The following packages are available:
  1  SUNWramd     Solaris 2.0 Example Ramdisk Driver
                  (sparc) 1

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1

Processing package instance <SUNWramd> from </dev/rmt/0h>

Solaris 2.0 Example Ramdisk Driver
(sparc) 1

This product is copyright by me.
Using </opt> as the package base directory.
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with
super-user
permission during the process of installing this package.

Do you want to continue with the installation of this
package [y,n,?] y

Installing Solaris 2.0 Example Ramdisk Driver as <SUNWramd>
```

Figure 3-10 Sample Run of `pkgadd`

```
## Installing part 1 of 1.
/opt/SUNWramd/bin/testram
/opt/SUNWramd/src/Makefile
/opt/SUNWramd/src/ramd.c
/opt/SUNWramd/src/testram.c
/opt/SUNWramd/usr/include/sys/ramio.h
/opt/SUNWramd/usr/kernel/drv/ramd
/opt/SUNWramd/usr/kernel/drv/ramd.conf
[ verifying class <none> ]
## Executing postinstall script.
Installing driver and configuration file in
/usr/kernel/drv.

Installation of <SUNWramd> was successful.
```

Figure 3-10 Sample Run of `pkgadd` *(Continued)*

To remove packages, use `pkgrm`. To run `pkgrm`, type:

```
# pkgrm <pkgname>
```

where `<pkgname>` is the name of your package.

The following is a sample run of `pkgrm`.

Code Example 3-1 Sample Run of pkgrm

```
# pkgrm SUNWramd

The following package is currently installed:
   SUNWramd        Solaris 2.0 Example Ramdisk Driver
                   (sparc) 1

Do you want to remove this package [y,n,?,q] y

## Removing installed package instance <SUNWramd>

This package contains scripts which will be executed with
super-user
permission during the process of removing this package.

Do you want to continue with the removal of this package
[y,n,?,q] y
## Verifying package dependencies.
## Processing package information.
## Executing preremove script.
```

Code Example 3-1 Sample Run of pkgrm (Continued)

```
The following files are being removed.
/dev/dsk/ramd0
/dev/rdsk/ramd0
/usr/kernel/drv/ramd
/usr/kernel/drv/ramd.conf
## Removing pathnames in <none> class
/opt/SUNWramd/usr/kernel/drv/ramd.conf
/opt/SUNWramd/usr/kernel/drv/ramd
/opt/SUNWramd/usr/kernel/drv
/opt/SUNWramd/usr/kernel
/opt/SUNWramd/usr/include/sys/ramio.h
/opt/SUNWramd/usr/include/sys
/opt/SUNWramd/usr/include
/opt/SUNWramd/usr
/opt/SUNWramd/src/testram.c
/opt/SUNWramd/src/ramd.c
/opt/SUNWramd/src/Makefile
/opt/SUNWramd/src
/opt/SUNWramd/bin/testram
/opt/SUNWramd/bin
/opt/SUNWramd
## Updating system information.

Removal of <SUNWramd> was successful.
```

Package Utilities

You may find the following utilities useful in managing software packages.

pkgchk

The `pkgchk` utility checks the accuracy of a package before or after installation.

To verify the accuracy of a package on a floppy disk, type:

```
# pkgchk -d /dev/diskette
```

To check the accuracy of a `pkgmap` file, type:

```
# pkgchk -m <pkgmap>
```

where `<pkgmap>` is the name of your `pkgmap` file.

The pkgchk utility is silent unless it finds errors or you specify the -v or -l flag.

For more information, see the man page for pkgchk.

pkgparam

For information about packages on the system, use the pkgparam utility to display a package's parameter values from its pkginfo file.

For information about an installed package, type:

```
# pkgparam -v <pkgname>
```

where <pkgname> is the name of your package.

The -v (verbose) flag causes pkgparam to print the parameter names; without the -v flag, pkgparam prints only the parameter values.

For information about a package on a floppy disk, type:

```
# pkgparam -v -d /dev/diskette <pkgname>
```

where <pkgname> is the name of your package.

pkgask

The pkgask utility stores answers to a request script for a package.

To run pkgask, type:

```
# pkgask [-d device] [-R root_path] -r response pkginst1 [pkginst2 ...]
```

where:

device(optional)	runs the request script for a package on device. The device can be a directory pathname or an identifier for a tape, floppy disk, or a removable disk. The default device is the installation spool directory.
root_path (optional)	defines the full pathname of a subdirectory to use as the root path.
response	identifies a file or directory which should be created to contain the responses to interaction with the package. The name must be a full pathname.

`pkginst1,pkginst2...` specifies either the package instance or the list of instances for which request strips will be created; it may be used to refer to all packages available on the source medium.

pkginfo

The `pkginfo` utility give you information about packages already installed on the system or some other media. To run `pkginfo`, type:

```
# pkginfo
```

with no arguments.

For information on optional arguments to `pkginfo`, see the man page for `pkginfo`.

Bibliography

SunOS 5.0 Application Packaging and Installation Guide, Part Number 800-6347-10, SunSoft, Inc., Mountain View, CA