Software Mastering Toolkit Guide

The Integrated Open Systems Environment

Date: 24 June 1992
Document version: 2.0.0A

Preface

The SCO Software Mastering Toolkit (SMT) permits files to be installed from a disk or tape using the SCO UNIX **custom**(ADM) utility. While installing files from disk or tape, called the "distribution media," **custom** provides additional services such as creating files and executing programs. The complete procedure of installing and creating files, and executing programs is called an "installation."

About this guide

This guide is for developers who are preparing applications or device drivers for installation, or who are porting applications or device drivers from MS-DOS® or other forms of the UNIX Operating System to SCO UNIX System V.

This guide assumes the following:

- You are familiar with the use of your operating system.
- You are familiar with the principles of shell programming.
- You read the documentation supplied with your system.
- You installed the SMT package with your Development System.

Contents of this guide

Here is a brief description of each chapter in this guide:

- Chapter 1, "Introduction," provides an overview of software mastering and the tools included in the SMT.
- Chapter 2, "Stocking the source tree," describes how to arrange your product source files into a single source directory. The chapter also provides guidelines for writing installation scripts.
- Chapter 3, "The site_variables file," explains how to create a *site_variables* file that describes how to cut your distribution.
- Chapter 4, "Creating the distribution directory," explains how to use the **docut** utility to generate a distribution tree and permlist, and, if no additional changes are required, cut the distribution.
- Chapter 5, "Modifying the distribution," explains how to define packages in your permlist and how to add and remove files in your distribution.
- Chapter 6, "Cutting the distribution," describes how to cut the distribution on the distribution media.
- Chapter 7, "Verifying the distribution," explains how to revise your product source and distribution files.
- "Software Mastering Toolkit utilities" contains manual pages for all of the utilities explained in this guide.
- A glossary and an index are provided at the end of this guide for your reference.

Product features

The SMT provides several features that can be useful for optimizing how your distribution is produced. These features are:

- **Compressing files**

  You can create a compressed distribution to conserve distribution media.

- **Disk image copying**

  When making multiple copies of your software, you can create a single file for each distribution floppy disk to be copied. Because each file is the size of the target floppy disk, it can be copied easily to make the distribution copies.

- **Packages**

  You can organize your product into units corresponding to pieces of your product, tasks, or use any size of your choosing. **custom** recognizes these "packages" as products that can be installed individually.

- **Permissions list**

  The SMT automatically creates the permissions list for your distribution. This file instructs the **custom** utility during an installation to direct where files and directories are located, assigns permissions, and performs other tasks such as creating device nodes. System administration utilities use this list to determine which products are installed and their associated files.

- **Sum list**

  You can create a sum list at the end of the distribution so that you can readily tell if any changes were made between copies of the distribution.

Conventions used in this guide

SCO documents use font changes and other typographic conventions to distinguish text elements. The following table shows these conventions:

Font conventions

Example	Entity
cc or **cc**(CP)	command. The "CP" indicates the manual page section in which the command is documented.
libc.a	filename
action	placeholder
⟨Esc⟩	keyname
open or **open**(S)	system calls, library routines, kernel functions, C keywords. The "S" indicates the manual page section in which the command is documented.
buf	structure
`b.errno`	structure member
$HOME	environment or shell variable
SIGHUP	named constant
`login`	output
"clobber"	jargon
"adm3a"	data value
date	user input

Chapter 1

Introduction

The SCO Software Mastering Toolkit (SMT) makes an application installable with the **custom**(ADM) utility.

When **custom** installs a software product, the files that comprise the product are transferred from the media on which the software product is distributed to the computer on which the product is being installed. Using the information supplied in a permissions list, **custom** places the files in the correct locations, assigns permissions, and executes scripts that perform optional tasks such as system or product configuration.

Components of a custom-installable product

The SMT allows you to distribute your software in a **custom**-installable product. It creates your final product from three components:

source files comprise the portion of your software used after being installed on the target system

permissions list ("permlist")
lists the specific location of each file that can be installed on the target system. The SMT generates a permlist for you.

installation scripts (optional)
manage the installation at any of three stages:

- before files are extracted from the distribution media
- after files are extracted from the distribution media
- before removing files from the target system

Roadmap to software mastering

The following steps describe how to develop a **custom**-installable product with the SMT. Software mastering usually involves revising an initial cut and different projects require different types of maintenance.

1. Stock the *./source* tree.

   The SMT builds your product from files that reside in a directory reserved exclusively for your product, referred to throughout this guide as the ***prd*** directory. Build your source directory as completely as possible. If final files are not available, use dummy files; it is easier to replace files in your source tree than to add or remove them. For details, see Chapter 2, "Stocking the source tree."

2. Define the product.

   The manner in which your product's files are placed on the distribution media is defined by the contents of a file named *site_variables*, which resides in your ***prd*** directory. For details, see Chapter 3, "The site_variables file."

3. Generate the distribution tree.

   Use the SMT tools to read your product source tree and create a parallel distribution tree from which you can cut your distribution diskettes or write a disk image file. The SMT tools automatically create a permlist for you. For details, see Chapter 4, "Creating the distribution directory."

4. Create **custom**-installable packages (optional).

   If you want your software to be installable in packages, edit the permlist to specify the names and contents of packages. For details on creating packages for **custom**, see Chapter 5, "Modifying the distribution."

5. Write the distribution to the desired media.

   You can either cut your distribution on the selected media or generate a "disk image" file that contains a binary image of the data as it will appear on the distribution media. You can later write the disk image to the distribution media. For details, see Chapter 6, "Cutting the distribution."

6. Verify the distribution cut.

   After cutting your distribution media, verify that the cut was successful by installing your product on a system. For details, see Chapter 7, "Verifying the distribution."

Toolkit software

This section briefly introduces the SMT utilities and describes their relationships to each other as well as to **custom**. These utilities are described in the manual pages in Appendix A.

You run the following utilities to perform software mastering:

custom(ADM) installs a product distribution. Using the permlist, **custom** extracts the files from the distribution media, such as a floppy disk, to the system on which the product is being installed. A permlist is a file that describes where each file in the distribution should be located when the files are extracted to the computer on which the software product is being installed. The permlist also describes the permission modes that are assigned to each file as it is installed. The **mkperm**(SMT) utility creates the permlist.

fixperm(ADM) corrects file permissions and ownerships in a distribution. Using the permlist, **fixperm** resets permission modes back to the value designated in the permlist. **docut** and **mkcuts** call this utility.

docut(SMT) creates a **custom**-installable distribution. Always run **docut** first to prepare the distribution automatically. This utility prompts you for information required to find and create a distribution. The **docut** utility executes the **mkperm** utility to create a permlist.

mkflops(SMT) creates output floppy disks using disk image files generated by **mkcuts**(SMT) or **diskimage**(SMT)

diskimage(SMT) reblocks the distribution files into a file that is the size of the output disk media so that the file is copied directly onto a floppy disk using the **mkflops** utility. **mkcuts**(SMT) calls this utility.

hocheck(SMT) compares a permlist against the actual files in the distribution directories. Filenames containing discrepancies are noted and placed in a directory. The permlist is created with the **mkperm** utility and can be edited with a text editor such as **vi**(C). You must ensure that changes to the file are not overwritten when other SMT utilities are run. **hocheck** ensures that the permlist is correct.

The following utilities are provided for use by the SMT programs; you do not run these utilities directly.

fdfit(SMT) fits the distribution files onto the floppy disk volumes in the most efficient manner for each file in the permlist. This utility ensures that a file is not split between floppy disks, unless the size of the file exceeds the size of a floppy disk. **docut** and **mkcuts** call this utility.

mkperm(SMT) generates a permlist for all files in the distribution. The permlist includes all files in the distribution along with their file permissions and relative pathnames. Used during installation, the list controls which files from your distribution get placed onto media. **docut** calls this utility.

permlint(SMT) checks the syntax of the permlist

pkgsize(SMT) updates the total disk space required by each package listed in the permlist. A package is a logical portion of the entire application (help files or font files, for instance). A package lets you selectively choose which portions of the application to install with **custom**. After running **pkgsize**, you can go back to the permlist and find the correct byte size listed for each package name. This tells you how much disk space each package in your distribution uses. **docut** and **mkcuts** call this utility.

volno(SMT) updates the permlist to select the file-to-volume allocation determined by **fdfit**(SMT). After running this utility, you can go back to the permlist and find a volume number listed beside each file that tells you on which volumes your files are located. **docut** and **mkcuts** call this utility.

SMT sample files

The SMT includes a set of sample projects with which you can test the SMT tools. The sample projects reside in the following directories under */usr/lib/scosmt*:

sample1
sample2
sample3
sample4
sample5
sample6
sample10
sample11
sample12
tapeinst

Samples 1-6 demonstrate simple software mastering techniques. Samples 10-12 demonstrate how to set up a project development environment. The "tapeinst" sample project demonstrates how to cut a product to tape.

Chapter 2

Stocking the source tree

Before you use the SMT utilities, it pays to spend time planning the specific contents of your product. You can spare yourself hours of file editing simply by giving careful consideration to the pathnames that will be created when your product is installed on target systems. The first half of this chapter describes the file location conventions used by the SMT, and how to prepare your product source files before using SMT.

For many projects, it may be sufficient to simply install your product's files, overwriting any existing files of the same name. For other projects, you may need to include installation scripts that tailor the installation to the requirements of the target system. The second half of this chapter provides guidelines on how to include installation scripts in your distribution so that **custom** automatically executes them at the appropriate time.

Organizing source files

Because the files of a product may be installed (and removed) according to package organization, they should be grouped in a logical pattern that allows you maximum flexibility in the installation. Similarly, packages should be grouped in logical patterns within each product.

Several conventions concern file locations:

- **Store all front-end shell script files that are accessed directly by the user in */usr/bin*.**

  A front-end shell script contains code that calls binary and other shell script files. The binary files that are called should be stored in */usr/lib/prd*. Binary files should not be stored in */usr/bin*. Keep the number of the files placed in */usr/bin* to a minimum because, if this directory gets too full, it can slow the system down. If possible, all the files should be relocatable. If you use an alternate location, indicate the new location with a shell environment variable.

- **Store all supporting files in special directories located in */usr/lib/prd*, where *prd* identifies the product.**

  For example, supporting files for SCO Professional are located in the subdirectory */usr/lib/pro*. If possible, these files should also be relocatable.

- **Store any sample files written for demonstration or educational purposes, such as tutorials, in the application's library directory.**

  Files such as these, and information files that are temporary, can be grouped together in a separate package that is optionally installable. This way, information is available to the user, but is removable if the user wants to conserve more disk space.

Setting up a distribution hierarchy

Perform the following steps before using **docut**:

1. Create a directory that is named after your product. The directory name must consist of no more than eight lowercase alphanumeric characters. Throughout this guide, this directory is referred to as *prd*. The *prd* directory can reside anywhere in your system.
2. Create the *prd/source* directory.
3. Copy your product source files into the *prd/source* directory, creating subordinate directories as required. For example, if your product contains a file that must be installed as */usr/lib/filename* on the target system, copy *filename* into *prd/source/usr/lib*.
4. Copy the init script(s) to the *./source/tmp* and the prep script(s) to the *./source/tmp/perms* directories, and copy the removal script to the *./source/usr/lib/custom* directory. For details on writing installation scripts, see the "Init, prep, and removal scripts" section in this chapter.

   NOTE If your product requires installation scripts, but you have not yet written them, create stub scripts in the appropriate locations. You can use the SMT before your files are finished, but it is easier to replace the stub scripts than it is to add new files.

5. If you intend to use **docut** to cut your distribution media, ensure that you have floppies or other media on hand before you run **docut**. The floppies need not be formatted; you can format them while running **docut**.
6. Log in as *root*. You must be *root* to use **docut** or **mkcuts**.
7. Make sure that every file has a path that corresponds to its destination. For example, if you have a file called *gizmo* that should be placed in */usr/bin* when your product is installed, put the file in *./source/usr/bin/gizmo*.

8. Make sure that the permissions and ownership are correct for every file.
9. Display the */etc/default/tar* file to determine the size of each distribution volume, its blocking factor, and the device name. Use any display command, such as **more**(C) or **cat**(C). For example, a few lines of a sample */etc/default/tar* file are as follows:

```
#     device                    block   size    tape
archive0=/dev/rfd048ds9         18      360     n
archive1=/dev/rfd148ds9         18      360     n
archive2=/dev/rfd096ds15        10      1200    n
archive3=/dev/rfd0135ds18       18      1440    n
archive4=/dev/rct0              20      150000  y
```

   From this listing, for a 1.2-Mbyte floppy disk (archive2), the size is 1200 kilobytes, the blocking factor is 10, and the device name is */dev/rfd096ds15*. These values should be put into the variables **DEVICE**, **BLOCKING**, and **VOLSIZE** in the *site_variables* file.
10. Change into the directory that contains *./source*, using **cd**(C).

Init, prep, and removal scripts

During installation, **custom** runs three special-purpose scripts:

PREPARATION SCRIPT (prep)
: runs before any other installation activity

INITIALIZATION SCRIPT (init)
: runs immediately after extracting the files from the media

REMOVAL SCRIPT (rmv)
: runs when a product is removed

Initialization, preparation, and removal scripts must be placed in the correct locations in the distribution hierarchy. Refer to the next section, "Writing installation scripts," for more information on the scripts described in this section. In addition, refer to the sample scripts supplied with the Toolkit software:

/usr/lib/scosmt/sampleX/source/tmp/init.sampleX
/usr/lib/scosmt/sampleX/source/tmp/perms/prep.sampleX
/usr/lib/scosmt/sampleX/source/usr/lib/custom/sampleX.rmv

Here *X* is the number of a sample directory.

Additional sample scripts are provided for installable device driver (idd) products.

Follow these guidelines to place the init, prep, and removal scripts in their correct locations:

- **Initialization script**

  Init scripts are run after package files are extracted. Multiple initialization scripts are permitted for a product. The init scripts should go in the *./source/tmp* directory. The name may be of the form *init.pkgname*. If your product is not divided into packages, the name should be of the form *init.prd*, where *prd* is the abbreviated product name.

- **Preparation script**

  Prep scripts are run before any product files are extracted. One preparation script is permitted for a product. The prep script should go in the *./source/tmp/perms* directory. The name should be of the form *prep.prd*, where *prd* is the abbreviated product name.

- **Removal script**

  One removal script is permitted, but not required, for a product. The removal script is placed in the *./source/usr/lib/custom* directory. The name must be of the form *prd.rmv*, where *prd* is the abbreviated product name.

Writing installation scripts

Several conventions are established for the UNIX system for consistency. These conventions are adopted by SCO and are presented here as suggestions to developers.

SMT script library

Design your script USER INTERFACE to be similar to that of the UNIX system and SCO applications to provide a consistent appearance and familiar interactive features. SMT includes standard functions to accomplish this. For example, the **getyn** function prompts the user for a yes or no answer.

The SMT includes a set of script functions that you can include in your installation scripts. Use these functions to provide a user interface that is consistent with the SCO installation scripts without writing them from scratch. The script libraries reside in */usr/lib/scosmt/scriptlib* and are well-documented in the script comment lines.

Bourne shell programming guidelines

The following are some Bourne shell programming guidelines to aid in your development:

- Use variables to make your script more legible (for example, exit values).
- Define variables explicitly at the start of the program (for example, **PATH**).

- Be aware of the output of programs invoked; direct errors to *stderr*, and direct extraneous output to */dev/null*.
- Use a colon (:) as the first character of the first line to ensure execution as a Bourne shell script (as opposed to a *csh* script).
- Always specify an explicit exit value (0=success; 1=failure).

Writing an initialization script

Create an initialization script for any special installation requirements, such as relinking the kernel or prompting the user for system-specific information. The init script is run during the installation process, right after the files are extracted from the media.

Here are a few typical procedures to include in an init script:

- Print the copyright message.
- Customize files and configuration after installation has taken place.
- Execute product runtime configuration.

Follow these steps to create an init script:

1. Create the init script in the *./source/tmp* directory.
2. Name the init script using this syntax:

   init.prd

   Here ***prd*** matches the product variable listed in the permlist. (***prd*** is also called the **prd** value, which is a variable listed in the permlist, and is the same value as **PRODPRD** in *./site_variables*.) Or, if there are init scripts unique to each package, use this syntax:

   init.pkgname

   Here ***pkgname*** matches the package name.
3. Make sure that a zero (0) exit status is returned for successful initialization and a one (1) is returned in cases of failure, because the **custom**(ADM) utility checks the exit status of the initialization script.
4. When setting permissions for the init script, make sure that it is executable by **custom**, and the owner and group are both *bin*. Set the permissions and ownership with these commands:

   chmod 755 init.*pkgname*
   chown bin init.*pkgname*
   chgrp bin init.*pkgname*

5. This step is necessary only if you run **mkcuts**(SMT) as a standalone utility. If you plan to use **docut**(SMT) to cut the distribution, then you can skip this step.

Add a line to the permlist for the init script file, listing the init script in the same package as the files to which it is associated. For example, if your product has three packages, only one of which has an init script, then list the init script in only that package in the permlist. In this case, if the user chooses to install a single package, only the initialization script associated with that one package is invoked.

Conversely, if you have a separate init script for each of your three packages, then these scripts must still be distributed in *./tmp*, each must have a unique name in the form *init.pkgname*, and each must be listed in the package with which it is associated.

Finally, if you want the same init script to be run with every package in your distribution, it must be listed in every package in the permlist.

Writing a preparation script

Create a prep script for an activity that must take place before files are read from disk, such as saving the users' files before updating a product. The prep script is run only once: when the product is first installed, before any of the files (other than the permlist) are extracted from the media.

Prep scripts must be placed in the PERM package subdirectory.

Here are a few typical procedures to include in a prep script:

- Check for pre-existing files to avoid overwriting.
- Check the version of the operating system or the CPU to determine if the target is acceptable.
- Ask the user miscellaneous questions before installation.
- Ask the users if they really want to overwrite the older version of the application that already sits on the hard disk.

Follow these steps to create a prep script:

1. Place the prep script in the *./source/tmp/perms* directory.
2. Name the prep script using this syntax:

   prep.prd

   Here *prd* matches the product code name (the **prd** variable listed in the permlist, or **PRODPRD** in *./site_variables*).
3. When setting permissions for the prep script, make sure that it is executable by **custom** on the distribution as well as in the permlist.

4. Make sure that a zero (0) exit status is returned for successful initialization and a one (1) is returned in cases of failure, because the **custom** utility checks the exit status of the prep script.
5. The following step is necessary only if you run **mkcuts** as a standalone utility. If you plan to run **docut** or **mkperm**, then you can skip this step.

   Add a line to the permlist for the prep script file. Make sure that it gets listed in the PERM package. (Refer to step 5 in the section "Writing an initialization script" in this chapter.)

Writing a removal script

Create a removal script if one of the following two cases is true:

- Some of the files associated with the application are not listed in the permlist (for example, they are created during installation) and, therefore, cannot be removed automatically by **custom**.
- Some special activity done in the init script, or some other setup script, needs to be "undone," such as adding information to a system file or changing a system parameter.

In either case, the removal script is similar to the init script, except that it is run when the **custom** utility's "Remove one or more packages" option is selected.

When writing your removal script, remember that the removal script is run before any of the installation files are actually removed.

For example, your script could remove */etc/rc* lines added when the product is installed, such as the remove file recovery code from */etc/rc*. Under the UNIX system, installation scripts are in the */etc/rc.d* directory.

Follow these steps to create a removal script:

1. Place the removal script in the directory *./source/usr/lib/custom*.
2. Name the removal script using this syntax:

   prd.rmv

   Here *prd* matches the product variable listed in the permlist. (*prd* is also called the **prd** value, which is a variable listed in the permlist, and is the same value as **PRODPRD** in *./site_variables*.)

   NOTE While a shell script is not the required format for the rmv script, it is recommended because shell scripts are powerful, portable, and easy to maintain.

When setting permissions for the rmv script, make sure that it is executable by **custom** on the distribution as well as in the permlist.

3. This final step is necessary only if you run **mkcuts** as a standalone utility. If you plan to run **docut** or **mkperm**, then you can skip this step.

   Add a line to the permlist for the removal script file.

A sample removal script can be found in the standard UNIX Extended Utilities set under */usr/lib/custom/unixos.rmv*. The Extended Utilities removal script is a good example of how a single removal script can perform different actions based on which package is being removed.

Checklist

Now that you have written the scripts required for your distribution, you are ready to run the utilities described earlier in this guide. These utilities actually prepare your permlist, which includes listings for the scripts prepared here for installation. Before continuing, you may want to use the following checklist to ensure that you have completed the scripts as accurately and completely as possible:

In writing the init script, did you complete the following tasks?

- ❒ Write an init script for special installation requirements, such as relinking the kernel.
- ❒ Copy the init script into *./source/tmp* and name it with the syntax *init.prd* or *init.pkgname*. If your product is not divided into packages, the name should be of the form *init.prd*, where *prd* is the abbreviated product name (also called the **prd** value, which is a variable listed in the permlist, and is the same value as **PRODPRD** in *./site_variables*).
- ❒ Within the script, make sure to return a zero (0) status for successful installation and a one (1) status in case of failure.

In writing the prep script, did you complete the following tasks?

- ❒ Write a prep script for activities that must take place before installation, such as saving the users' files before updating the product.
- ❒ Place the prep script in *./source/tmp/perms* and name it with the syntax *prep.prd*.

In writing the removal script, did you complete the following tasks?

- ❒ Write a removal script for special removal requirements, such as undoing an activity performed by the init script or removing files that are not listed in the permlist. Write the script so that it returns an exit status of 0 or 1.
- ❒ Place the removal script in */source/usr/lib/custom* and name it with the syntax *prd.rmv*.

In writing each of the scripts, did you remember the following?

- ❒ Set *root* permissions to read and execute.

Chapter 3

The site_variables file

The **mkcuts**(SMT) utility uses a set of variables to which you can assign values. To set the variables:

1. Copy the */usr/lib/scosmt/site_variables* file to the ***prd*** directory.
2. Set the *site_variables* file permissions to 644 with the following command:

   chmod 644 site_variables

3. Make the changes to *site_variables,* using the descriptions in this section and the comments in the file as a guide. The file contains the variables listed here in alphabetical order:

BASEPACKDESC	PERMLIST
BASEPACKDIR	PRODPRD
BASEPACKNAME	PRODREL
BLOCKING	PRODSET
CDISTDIR	PRODTYP
DEVICE	PRODUPD
DISTDIR	SERIALIZE
FORMAT	SERIALKEY
IMAGEDIR	SUMDIR
MISCDIR	TARGET_OS
NOCOMPRESS	VOLPREFIX
NOTAR	VOLSIZE
OTHER_MEDIA	

Description of site_variables file variables

PRODSET Enter the full name of your distribution's product. Use quotations marks (" ") around a name that has more than one word. Although the string can contain up to 99 characters, keep this name to a size that **custom** can display without truncating. The permlist is modified to match this variable. Use the following syntax:

PRODSET="*full_product_name*"

PRODPRD Enter a short, or abbreviated, product name. The permlist is modified to match this variable. Use the following syntax:

PRODPRD=*shortened_product_name*

shortened_product_name must be a unique string of no more than 8 lowercase alphanumeric characters that matches the name of your product (***prd***) directory. Do not include spaces in the string.

PRODTYP Enter the code of the target machine and environment type. Refer to the "Supported machines and environments" section in this chapter for a list of the target system variables. The permlist is modified to match this variable. Use the following syntax:

PRODTYP=*target_machine_and_environment_type*

PRODREL Enter the current release number and iteration letter. The permlist is modified to match this variable. Use the following syntax:

PRODREL=*current_release_number*

TARGET_OS Enter the operating system on which the product will run. It must be either "UNIX" or "XENIX". Use the following syntax:

TARGET_OS=*operating_system*

BASEPACKDIR If the product has only one major package, enter the directory name where the source files reside. Use the following syntax:

BASEPACKDIR=*directory*

BASEPACKNAME If the product has only one major package, enter the package name here. Use the following syntax:

BASEPACKNAME=*package_name*

SCO's convention is to use all uppercase characters.

BASEPACKDESC If the product has only one major package, enter the full description of the package here. This should be a quoted string.

BASEPACKDESC="*full_package_description*"

DEVICE Enter the name of the device that is archiving the distribution. Refer to the */etc/default/tar* file for full listings of device names. Use the following syntax:

DEVICE=*archiving_distribution's_device_name*

VOLSIZE Enter the byte size of each distribution volume. Here are the most commonly used device names and sizes:

Device	Volume size
/dev/rfd096ds15	1200K (1.2 Mbytes)
/dev/rfd096ds9	360K
/dev/rfd0135ds9	720K
/dev/rfd0135ds18	1440K (1.4 Mbytes)

Use the following syntax:

VOLSIZE=*distribution_volume_size*

The number used here depends on the size of one media volume on which the product is distributed. Suffixes after the number multiply the number accordingly: **k** multiplies the number by 1024; **b** by 512; **w** by 2.

NOTE Be sure to include a suffix, or ***distribution_volume_size*** is interpreted as the number of bytes per volume.

BLOCKING Enter the blocking factor of the archiving device. This number represents the number of blocks that are read from or written to the device each time a read or write is requested from a user program. Refer to the */etc/default/tar* file for this value. Set the **BLOCKING** variable, if required, using this syntax:

BLOCKING=*blocking_factor*

FORMAT Enter the complete command needed to format a volume. Make sure to use quotation marks (" ") around the command. For the **FORMAT** variable, use the following syntax:

FORMAT="*complete_format_command*"

NOTAR If in **mkcuts** you wish to hold back some files from being copied to the output media, set **NOTAR** to the names of the files. Use this syntax:

NOTAR=*pathnames*

To specify a single file, ***pathnames*** is a single pathname. To specify multiple files, ***pathnames*** is a quoted list of pathnames, delimited by linefeeds.

NOCOMPRESS Enter the names of any packages that are not to be compressed when the **-c** option is used with **docut** or **mkcuts**. If this is not set, the only file not compressed is the permlist. The string must be quoted if there is more than one package name. Use the following syntax:

NOCOMPRESS="*package_name(s)*"

PRODUPD Enter the update string, if applicable. This string is two characters that must start with "U" (for update). Use the following syntax:

PRODUPD=*update_string*

If you are not cutting an update, it is crucial that you leave this field blank.

SERIALKEY
SERIALIZE These two variables are reserved for future use.

PERMLIST Enter the name of a directory where the permissions list is stored if other than the default path (*./tmp/perms/PRODPRD*). The syntax is:

PERMLIST=*pathname*

DISTDIR Enter the name of the directory where the distribution files are placed for use by **mkcuts**. The syntax is:

DISTDIR=*pathname*

The default is *./dist*.

MISCDIR Enter the name of the directory where temporary files are stored by the SMT utilities:

MISCDIR=*pathname*

The default is *./misc*.

CDISTDIR Enter the name of a directory where the compressed distribution files are to be stored. For more information, refer to the "mkcuts -c (compress) option" section in Chapter 6, "Cutting the distribution." The syntax is:

CDISTDIR=*pathname*

The default is *./cdist*.

IMAGEDIR Enter the name of the directory from which the disk image directory hierarchy is created. For more information, refer to the "mkcuts -i (disk image) option" section in Chapter 6, "Cutting the distribution." The syntax is:

IMAGEDIR=*pathname*

The default is "`pwd`/cut".

SUMDIR Enter the name of the directory where the sums list is stored. For more information, refer to the "mkcuts -s (sums list) option" in Chapter 6, "Cutting the distribution." The syntax is:

SUMDIR=*pathname*

The default is "`pwd`/cut".

If you want to make changes to any of the values that you set up, edit the appropriate parameter file.

Supported machines and environments

The following are supported machines and environments and the corresponding **PRODTYP** values to use in the *site_variables* file.

Type	Description
k286	For n286 machines, specific to the UNIX/XENIX kernel
k386	For n386 machines, specific to the UNIX/XENIX kernel
ku386	For u386 machines, specific to the UNIX kernel
n286	For 286 or 386 machines running UNIX/XENIX
n286.5	Like n286, except must run UNIX/XENIX
n386	For 386 machines running UNIX/XENIX
n86	For any Intel-based 80*86 UNIX/XENIX
u386	For machines running SCO UNIX

Chapter 4

Creating the distribution directory

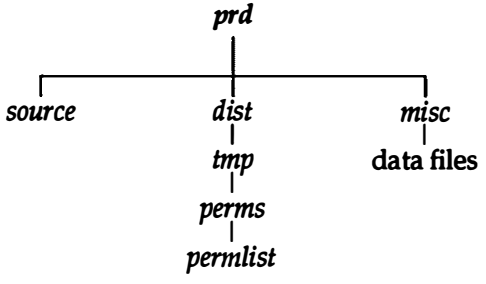
This chapter describes how to set up an initial distribution. It assumes that you have already stocked the source directory and created a *site_variables* file that describes how you want to cut your product. Your initial distribution will not be divided into packages, though you can add package definitions later by editing your product's permlist. For details on defining packages, refer to Chapter 5, "Modifying the distribution."

The **docut**(SMT) utility handles almost all of the tasks necessary to make a **custom**(ADM)-installable application out of a group of files.

Running the docut utility

Once you have copied your distribution files and directories into the *source* directory, you are ready to cut the distribution with the **docut** utility. **docut** creates distribution volumes that are **custom**-installable.

The **docut** utility uses the following directory hierarchy:



The **docut** utility creates the *dist* directory, the *misc* directory, and the permlist, ***prd/dist/tmp/perms/prd***. The *source* directory is one that you create before running **docut**.

To create a new distribution:

1. Change to your *prd* directory.

   For example, to run **docut** on the *sample3* sample product included with the SMT, change to */usr/lib/scosmt/sample3*.

2. Start the **docut** utility with the following command:

   docut

   A series of messages appear as **docut** creates the distribution directory. For example, the following are messages that appear while creating the *sample3* distribution directory:

```
Checking for symlinks...
Creating permlist...

Directory /usr/lib/scosmt/sample3/dist does not exist.
Creating directory /usr/lib/scosmt/sample3/dist
Creating distribution hierarchy...

Executing the mkcuts utility from docut: mkcuts

Creating SCO SMT Sample3: SCO products with man pages  Release: 1.0.0a
Setting permissions...
Updating package sizes...
         Pkg      Blks     Files    Links    Dirs     Devs     Upd
         PERM     4        1        0        0        0         no
         PRDBASE  76       11       0        1        0        yes
         ALL      80       12       0        1        0        yes
Creating contents listings...
Vol No of  Bytes    Bytes   Percent
 no files    used     left  efficient
  1   13     46080  1182720    3.75
tot   13     46080  1182720    3.75

12288 bytes of core allocated
Cutting master media for Product: sample3 1.0.0a
Creating ../misc/01 master...
Insert media volume 1 in device /dev/rfd096ds15
C)ontinue, F)ormat, A)bort or S)kip to summing:
```

   All by-products of **docut** are completed, such as the permlist and the copying of the files to the *./dist* directory.

3. Enter **c** to complete the distribution and have the files copied to the distribution media using **tar**(C).

4. To avoid copying the files to the distribution media, enter **a**. Select Abort if you are creating a disk image or if you need to define packages for your product. Then run **mkcuts** to complete the distribution. Refer to Chapter 6, "Cutting the distribution," for more information. Abort also abandons

the current volume and lets you skip to the next one. Abort makes it easy to skip to a specific volume if you need to recut individual volumes (this is permissible as long as the permlist and file sizes have not changed).

5. Enter **s** to generate a file that contains the byte sums of the new distribution files.
6. Enter **f** to format a floppy disk volume before archiving any files.

Choose the appropriate option, and press ⟨Return⟩. When all volumes are handled, **docut** exits.

You now have a distribution from which you can cut the distribution media. For details, see Chapter 6, "Cutting the distribution." If necessary, you can define packages for your product or change the contents of the distribution as described in Chapter 5, "Modifying the distribution."

Using the docut options

If you need to update your distribution, you must run **docut** again. If you want **docut** to ignore the existing permlist, use the **-e d** option:

docut -e d

If you want to compress the distribution files using the **compress**(C) command, add the **-c** flag:

docut -c

NOTE Only versions of **tar**(C) that have the C option can uncompress these files. Most UNIX systems do not support this option. If the target system cannot support this capability, do not use **docut -c**.

Chapter 5

Modifying the distribution

This chapter describes how to define packages for your product and how to add and remove files in your product's *source* directory after creating the *distribution* directory.

Editing the permlist

The following are situations in which you may want to edit the permlist manually:

- You want your product to be installable in packages.
- You need to change permissions of files, but it is inconvenient to set the ownerships and permissions in the source hierarchy.
- You need to add or remove files from the distribution.

Permlist file format

All permlists consist of the following elements:

header — first lines of the permlist in which parameters, such as the ***prd*** and ***release*** values, are set. The header is created automatically by **docut** and is modified rarely.

package descriptions — series of lines that describe the names and contents of each of the product's packages. The **docut** utility creates a permlist consisting of a single package with the same name as the product. If you want your product to be installable in packages, you must add the package definitions manually. For details, see "Defining packages" later in this chapter.

Each line in a permlist contains a combination of the following:

- a number sign (#) and a space at the beginning of the line to indicate a comment line
- a number sign and an exclamation mark (!) or a number sign and a letter from "a" to "z" at the beginning of the line to indicate a special keyword phrase. Only the keywords described in the next section are currently defined.
- non-blank lines that do not start with a pound sign that describe the product

The **custom**(ADM) utility parses the keyword phrases and description lines to determine what is being installed or, in the case of directories or device files, created. There is one item specification per line.

Permlist example (Part 1 of 2)

```
1       # Small OS Permlist
2       #
3       #prd=smallos
4       #typ=386GT
5       #rel=3.2.4
6       #set="The Very Small OS Product"

7       uid    root   0
8       uid    bin    2
9       gid    root   0
10      gid    bin    2
```

lines 1-6 These lines introduce the product. Line 3 is the product name, line 4 specifies the computer processor code, line 5 establishes the release number, and line 6 provides space for a product description. The statements are:

#prd The product name is a lowercase string with a maximum of eight characters. The **prd** value has the same name as the permlist file.

#typ vendor-defined machine environment string

#rel vendor-defined release string

#set used as a verbose product name for user interface display

lines 7-10 Specify the user ID (UID) and group ID (GID) that are used for the root and bin logins while the product is being installed. In each line, the first field indicates the ID type (either gid or uid), the second field contains the user or group name for the ID, and the third field is the corresponding numeric ID.

These values are used in the package descriptions (such as those shown in lines 16-43).

Permlist example (Part 2 of 2)

```
11      #
12      #!ALL  300    Entire Product
13      #
14      PERM      F644    bin/bin     1   ./tmp/perms/smallos   01

15      #
16      #!BASEPKG  256    Basic Package
17      #
18      # directory that is part of the required product
19      BASEPKG   d755    bin/bin     1   ./dev
20      # directory not part of the required product (D signifies)
21      BASEPKG   D755    bin/bin     1   ./tmp
22      # standard file
23      BASEPKG   x711    bin/bin     1   ./bin/cat           02
24      # standard file with 3 links
25      BASEPKG   x711    bin/bin     3   ./bin/cp            01
26                                        ./bin/ln
27                                        ./bin/mv
28      # special chmod bit set, see chmod(S)
29      BASEPKG   x2111   bin/backup  2   ./bin/df            01
30                                        ./etc/devnm
31      # shell script: note that read permissions are needed
32      BASEPKG   f755    bin/bin     1   ./usr/lib/mkdev/lp  02
33      # character device node
34      BASEPKG   c440    root/root   1   ./dev/auditr        21/0

35      #
36      #!DEVPKG    26    Development Package
37      #
38      # block device node
39      DEVPKG    b440    root/bin    1   ./dev/root          1/40
40      # linked device node
41      DEVPKG    c622    root/bin    3   ./dev/console       3/1
42                                        ./dev/syscon
43                                        ./dev/systty
```

line 14 — This statement specifies the location of the permlist file and establishes that it is to be included in the **PERM** package. The permlist is always assigned to this package and must reside in the */etc/perms* directory. On the distribution, the permlist is in *./tmp/perms*. **custom** moves the permlist into */etc/perms* during installation. Refer to the "File descriptions" section that follows for a complete description of the fields in this statement.

lines 16-43 — The rest of this permlist is hypothetical and provides descriptions for two packages, **BASEPKG** and **DEVPKG**.

Package declarations

Add-on application software is grouped into and installed as units called packages. A package is a logical group of files; a package groups files into functional units that can be installed as desired — either all packages together or individually. Multiple packages can be delivered on a single volume of media, or a package can span multiple volumes of media.

Each package must have a declaration containing the following fields: a package name, the package size in 512-byte blocks, and a string containing the package description. **custom** uses package declarations to present users a list of available packages. The only package that does not require a package declaration is PERM.

PERM is a reserved word. All products have a **PERM** package in their permlist that includes the files used when creating a product, except for the label file (which is not listed in the permlist). Before the **PERM** package declaration, the **#! ALL** statement is required. (Refer to lines 12 - 14 of the permlist example.) "ALL" is a reserved name that refers to the entire product.

Package names must be unique and can be up to eight characters in length.

File descriptions

A file description consists of the following tab-delimited data fields:

- package specifier
- permission
- owner specification
- group specification
- number of links on the file
- filename
- volume number

Refer to the permlist example, lines 14 through 43, for examples of file descriptions. The file description fields are:

Package specifier
: an uppercase, alphanumeric, non-white-space string, which is the name of a package within a distribution set.

Permission specification
a file type, followed by an octal **chmod**(C) permission specification. The file type can be one of the following characters:

Character	Meaning
x, X	Executable
a, A	Archive
e, E	Empty file
b, B	Block device
c, C	Character device
d, D	Directory
f, F	Text file
p, P	Named pipe
o, O	Okay. This indicates that there should be no file type checking, allowing any format or contents in what would normally be the header section of an executable. For example, data files should be of type "o".

An uppercase file type indicates that the associated file is optional. When **custom** installs a package, it provides a test to ensure that all pieces of the product or package is installed. If a package is not fully installed, **custom** lists a package or product as "Partially Installed." A product is considered fully installed if:

- the respective permlist file exists in */etc/perms*
- all required files are installed. A required file is indicated by a lowercase file type. If the letter is in uppercase, **custom** does not consider the file as part of the criteria for a complete installation.

  If all files are listed with uppercase file types, **custom** evaluates the product as installed only if the permlist for the product exists in */etc/perms*.

Owner and group specifications
owner and group permissions are in the third column separated by a slash: for example, *bin/bin*. The owner and group names are the same as those defined at the start of the permlist.

Links
number of links are in the fourth column. If there are links to the file, the following lines contain the linked filenames. Linked device nodes do not have volume numbers.

File pathname
fifth column is the pathname of the file on the distribution media volume and is relative to the root directory. The pathname must be relative (". /"); that is, it must not begin with a slash "/".

Volume numbers or major-minor numbers
sixth column has two uses:

- number of the media volume on which a file exists. This column is blank for directories because directories are created as needed when **custom** executes. The media volume has the same value as the *vol=* directory of the label file. The maximum number of digits in the volume number is two (up to 99 volumes are possible).
- In the case of special files, the major and minor device numbers are specified, separated by a slash. Device files are created by **custom** while it is executing. When a device file is linked to another device file, a value is not put in the sixth column of a linked device file entry. See lines 39 through 43 of the "Permlist example" for examples of device file entries.

Defining packages

When you create a distribution with **docut**, the resulting permlist contains only one package definition. This default package obtains its name and description from from the **BASEPACKNAME** and **BASEPACKDESC** variables in the *site_variables* file. If you want your product to be installable in packages, you must manually add package definitions to the permlist. For an example of a permlist that contains package definitions, see */etc/perms/scosmt*, the SMT permlist. For details on the permlist file format, see **fixperm**(SMT).

After adding package definitions to your permlist, you can cut your distribution as described in Chapter 6, "Cutting the distribution."

Adding and removing files

It is not uncommon to have to change the files in your distribution after already creating the distribution tree. In such situations, you have to run **docut** to update the distribution directory, and in some instances, you also have to edit the permlist.

Fortunately, **docut** detects if the new distribution tree is different than the distribution described in the old permlist. In these cases, **docut** asks if you want to generate a partial permlist that describes the changes. If you request **docut** to generate the partial permlist, it creates *prd/misc/partperms* instead of replacing the existing permlist. Sometimes it is easier to update a permlist from a *partperms* file, particularly when you have added a large number of files to the *source* directory and you have added package definitions to the permlist.

Adding files

This section describes how to add files to your *source* directory and update your product's permlist. Follow these steps if you only need to add a few files. If you need to add a large number of files, see "Adding files with a partperms file."

```
Checking that source and dist trees match...
./tmp/perms missing from source tree
Warning: the distribution tree is out of sync with
         the source tree. You may want to remove files
         from the dist tree to sync these trees up.
Do you want to remove any of these dist tree files ? (y/n) n
Checking for symlinks...
Creating permlist
Creating distribution hierarchy...

Comparing permlist to distribution tree.
Path to permlist is /usr/lib/scosmt/sample3/dist/tmp/perms/sample3
Path to dist tree is /usr/lib/scosmt/sample3/dist

Checking the distribution tree for extra or missing files...
All files in the permlist have been accounted for.
The distribution directory does not contain any extra files.

Maintaining previously existing permlist
since custom modifications can be kept.

Executing mkcuts.

Creating SCO SMT Sample3: SCO products with man pages  Release: 1.0.0a

Setting permissions...
Updating package sizes...
          Pkg       Blks      Files     Links     Dirs      Devs      Upd
          PERM      40        4         0         0         0          no
          PRDBASE   306       45        0         4         0         yes
          ALL       346       49        0         4         0         yes
Creating contents listings...
Vol No of  Bytes    Bytes   Percent
 no files    used     left  efficient
  1   20     71680  1157120    5.83
tot   20     71680  1157120    5.83

2 sets of linked files, 4 linked files total
15360 bytes of core allocated
Updating volume numbers...
Creating ../misc/01 master...
C)ontinue, F)ormat, A)bort or S)kip to summing: a
```

Adding files with a partperms file

Although it is easy to write a new entry in your permlist every time you add a file to the *source* directory, it may be impractical to do this when large numbers of files are added or removed. To eliminate the need for manual entries in the permlist, **docut** can generate a partial permlist that includes entries for files not already listed in the permlist.

To add files to your distribution using a partial permlist:

1. Add the new files to the *prd/source* directory.
2. Change to your *prd* directory using **cd**.
3. Start the **docut** utility with the following command:

   docut

   A series of messages appear as **docut** creates the *distribution* directory. For example, the following messages appear if you run **docut** from the */usr/lib/scosmt/sample3* directory:

```
Checking for symlinks...
Creating permlist...

Creating distribution hierarchy...

Comparing permlist to distribution tree.
Path to permlist is /usr/lib/scosmt/sample3/dist/tmp/perms/sample3
Path to dist tree is /usr/lib/scosmt/sample3/dist

Checking the distribution tree for extra or missing files...
All files in the permlist have been accounted for.

Extra files have been found in the dist tree...
Files stored in ./misc/extrafiles
```

   docut displays a list of the files you added to your *prd/source* directory and asks if you want to remove these files.

4. When you are asked if you want to remove any of the new files, type **n**, then press ⟨Return⟩.

The following messages appear:

```
WARNING: Your distribution tree has files that
are not present in your old permlist.
Constructing partial permlist...

A partial permlist containing these previously
un-encountered files has been stored in ./misc/partperm

If you want to maintain any custom
modifications you have previously made
you will need to hand modify
/usr/lib/scosmt/sample3/dist/tmp/perms/sample3

Abort any chance to modify the old permlist ? (y/n)
```

5. Type **n**, then press ⟨Return⟩.

   The following messages appear:

```
The permlist used to obtain this
information are in the ./misc directory:
old = sample3.old, new = sample3.new
Modify /usr/lib/scosmt/sample3/dist/tmp/perms/sample3 to effect
changes.
```

6. Check the new partial permlist, *./misc/partperm*.
7. Copy the contents of the partial permlist into your product's permlist, *./dist/tmp/perms/prd*.

   You can now cut your distribution to the media described in your product's *site_variables* file.

Removing files

To remove files from your distribution, take the following steps:

1. Delete from your product's permlist, *prd/dist/tmp/perms/prd*, all entries corresponding to the files you want to remove from the distribution.
2. Change to your *prd* directory using **cd**.

3. Start the **docut** utility with the following command:

   docut

   A series of messages appear while **docut** creates the distribution. For example, the following messages appear if you run **docut** from the */usr/lib/scosmt/sample3* directory:

```
Checking that source and dist trees match...
```

   docut displays the following message:

```
Checking for symlinks...
Creating permlist...

Creating distribution hierarchy...

Comparing permlist to distribution tree.
Path to permlist is /usr/lib/scosmt/sample3/dist/tmp/perms/sample3
Path to dist tree is /usr/lib/scosmt/sample3/dist

Checking the distribution tree for extra or missing files...
All files in the permlist have been accounted for.

Extra files have been found in the dist tree...
Files stored in ./misc/extrafiles
```

4. When **docut** asks if you want to remove any of the listed files, type **y**, then press ⟨Return⟩. The following message appears:

```
If you have removed files you must abort and re-run docut.
Abort procedure ? (y/n) y
```

5. Type **y**, then press ⟨Return⟩.
6. When your system prompt returns, run **docut** again.

   docut displays messages as it creates the new distribution based on the new permlist and *source* directory. Finally, the following prompt appears:

```
C)ontinue, F)ormat, A)bort or S)kip to summing: s
```

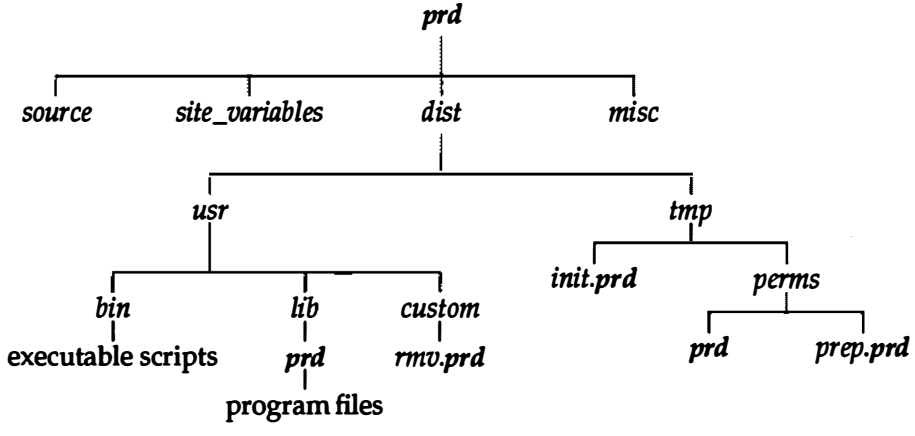
7. If you are ready to cut the new distribution to disk or tape, press **c** and press ⟨Return⟩. To make further changes or to use other SMT utilities, type **a** and press ⟨Return⟩.

Chapter 6

Cutting the distribution

This chapter explains how to use **mkcuts** to cut your distribution to disk or tape, or to create a disk image file from which you can duplicate diskettes with the **mkflops** utility later.

The **mkcuts** utility requires the following hierarchy:



Before running mkcuts

Before cutting a distribution using the **mkcuts** utility, follow these steps:

1. Ensure that you have floppy disks or other form of media on hand before running the **mkcuts** utility, because you will be prompted for them.

> **NOTE** If you are producing a product on tape, use 60M density, 9-track QIC24 tape cartridges of a quality equivalent to the 3M DC600a.

2. Stock the source tree as described in Chapter 2, "Stocking the source tree."
3. Create the *prd/site_variables* file as described in Chapter 3, "The site_variables file."
4. Create the distribution tree as described in Chapter 4, "Creating the distribution directory."
5. Define packages for your product by editing the permlist as described in Chapter 5, "Editing the permlist."
6. Log in as *root* and ensure that you have access to your distribution files and permissions list.
7. Make sure that you are in the *prd* directory.

Running mkcuts

Run the **mkcuts** utility, using the following syntax:

mkcuts *[option]*

The options are:

mkcuts option	Purpose
-c	creates compressed distribution files. This option requires that the **CDISTDIR** variable be defined in the *site_variables* file.
-i	creates a disk image
-s	creates a sums list
-f	specifies a *site_variables* file

The sections that follow show the messages and prompts that occur when you run **mkcuts**. The sections are listed as they appear in the program.

You can set implicit options for other tasks without specifying the command-line options through the use of variables that you set before running **mkcuts**. If the directories specified by these variables exist, prompts appear to which you enter **y** for yes, or **n** for no. These prompts remind you of the results of your previous **docut** session. The variables are:

Variable	Prompt
CDISTDIR	Do you wish compression?
IMAGEDIR	Do you wish to make images instead of floppies?
SUMDIR	Do you wish to make a sums list?

In each case, the directory to which the variable points must already exist.

Starting messages

The starting messages for **mkcuts** are:

```
Creating PRODSET PRODUPD Release PRODREL master.

Setting permissions...

Updating package sizes...
```

mkcuts -c (compress) option

The **-c** option uses the CDISTDIR variable, which contains the directory in which the compressed files are stored.

NOTE Only versions of **tar**(C) that have the C option can uncompress these files. Most UNIX systems do not support this option. If the target system does not support this capability, do not use **mkcuts -c**.

Thfis message is displayed:

```
percent% compression for: file
```

Then this message is displayed:

```
Setting permissions in compressed hierarchy...
```

If you respond with **n** to the previous compression prompt, **mkcuts** does not compress the directory.

More messages

The next set of messages is displayed:

```
Creating contents listings...

Creating MISCDIR...

Updating volume numbers...
```

mkcuts -i (disk image) option

The **mkcuts** utility creates a disk image that can then be copied to floppy disk using the **mkflops** utility. This option requires that the IMAGEDIR variable contain the name of an existing directory from which the image file directories are built. The actual directory in which the image file is stored is two levels lower than whatever name you specify. The directory names are derived from variables in the *site_variables* file so the final path to the disk image is IMAGEDIR/PRODREL/MEDIA. Even though this seems complex, the **mkflops** utility accesses this path directly if **mkflops** is called without arguments.
The MEDIA directory tests the value of the DEVICE variable and then translates it into the following values:

DEVICE Value	Resulting media-type
/dev/*48ds9	48ds9
/dev/*96ds8	96ds9
/dev/*96ds15	96ds15
/dev/*135ds9	135ds9
/dev/*135ds18	135ds15
/dev/*rct0	tape

If a device cannot be found in the **mkcuts** translation table, the image is placed in a directory named after the basename of the device.

If you are requesting a sums list in addition to the disk image, the sums information is stored in the *prd_sums* file. Refer to "mkcuts -s (sums list) option" later in this chapter for more information on a sums list.

When the **-i** option is selected, the following message is displayed to provide the image filename and directory path:

```
Creating disk image in directory/file
```

Final mkcuts prompt

If you did not specify the **-i** option, the following prompt is displayed:

```
C)ontinue, F)ormat, A)bort or S)kip to summing:
```

Enter **c** to **tar** the distribution files to your output media. You are prompted successively for each volume master in your distribution.

Enter **a** to stop processing for the current master. Abort makes it easy to skip to a specific volume if you need to recut a single volume (this is only viable as long as the permissions list and file sizes do not change).

Enter **f** to format a volume before archiving files. The string entered in the **FORMAT** variable is executed to format your media.

Enter **s** to not **tar** your distribution files to your output media. All other processing is performed, except for this aspect. Skip completes the current volume and lets you skip to the prompt for the next master in your distribution.

Choose an option, and press ⟨Return⟩. Continue to insert volumes as you are prompted for them.

You see this message:

```
Creating volume master...
```

When all volumes are handled, **mkcuts** exits.

mkcuts -s (sums list) option

This option creates a **sum**(C) **-r** checksum for each file in the distribution. This option depends on the **SUMDIR** variable being set in the *site_variables* file. The sums list pathname is **SUMDIR/PRODREL/MEDIA/***prd_sums*

The mkflops utility

This utility copies disk images to floppy disks or tape. Although **mkflops** has many options, the easiest way to run this utility is without options. Refer to the **mkflops**(SMT) manual page for more information about the options. Without any arguments, **mkflops** assumes that the image file or files are stored in the directory created by **mkcuts**, which has a pathname in this format:

IMAGEDIR/PRODREL/MEDIA

You can, however, run **mkflops** from a product directory or from the directory immediately above a product directory. This last option is particularly useful if you have a directory that contains several product directories because **mkflops** asks you to specify which product you wish to cut. Alternatively, you can specify a product with the **-p** option.

NOTE If you set the **FORMAT** variable in the *site_variables* file, note that this variable is not used in **mkflops**. The formatting options to **mkflops** prompt you to enter a floppy disk.

Chapter 7

Verifying the distribution

Now that you have cut your distribution, you are ready to test the installation procedure with the **custom** utility. This chapter describes both the installation and removal procedures using **custom**.

Using information generated by the SMT utilities, the **custom** utility installs the media volumes containing your distribution files. The utility checks to see that only the correct type of media is inserted, and also that the volumes are inserted in the proper order. Once **custom** has extracted the files, it runs the init script (if there is one). The init script may prompt the user for information. Then **custom** calls the **fixperm** utility to check the ownerships and permissions on the distribution files. Finally, the */tmp* files are removed, the action is logged in */usr/lib/custom/history*, and installation is complete.

The **custom** utility uses information generated by the **mkperm**(SMT) utility to do the following:

- create the point-and-pick lists displayed during an installation
- ensure that the media used matches the product being installed
- check that sufficient space is available on the hard disk before installing the package

custom invokes the **fixperm** utility to perform the following tasks:

- provide information about the installation status of files and packages in the product (whether or not they are installed)
- generate lists of files to be extracted by the **tar**(C) utility
- check that the correct files are installed, which includes verifying file types, ownership, and permissions

- create directories, zero-length files, devices, and file links
- prompt only for those volumes that contain files you have requested

Preparing for tape installation

Before you send out your tape distributions, be aware that **custom -t** always reads from tape via the same two device nodes: *dev/rct0* and *dev/xct0*. This can cause problems when installing tapes via **custom -t** because some tape drives are accessed via different device nodes, and unless such differences are resolved, **custom -t** will not let you install from tape. This is particularly important on systems equipped with multiple tape drives or with SCSI tape drives.

System administrators can modify their systems to allow **custom** to read any tape drive by creating links between the drive device nodes and */dev/rct0* and */dev/xct0*. For example, SCSI tape drives are accessed via */dev/rStp0* and */dev/xStp0*. To link these device nodes to the ones used by **custom -t**, execute the following commands:

ln /dev/rStp0 /dev/rct0 ln /dev/xStp0 /dev/xct0

You can then run **custom -t** to install products from tape.

Installing an application

All of the procedures described earlier in this guide led you to this point: making your application **custom** installable. This section describes the procedure for installing an application, or packages from an application, with the **custom** utility. With your distribution volumes in hand, and a machine that contains the **custom** utility, you are ready to test your installation.

1. Log in to the system as *root*.
2. If you plan to install from tape, be sure you have linked your tape device nodes to *rct0* and *xct0* as described previously in "Preparing for tape installation."
3. To install from floppy disk, type **custom** and press ⟨Return⟩.

   To select a particular disk drive, type the following command:

   custom -m */dev/name*

   The device names are listed in the */etc/default/tar* file.

4. To install from tape, type **custom -t** and press ⟨Return⟩.
5. The initial **custom** menu appears. From the list of choices, select the appropriate option; for an application, select Install, and press ⟨Return⟩.
6. You see the following message:

   ```
   Select a Product
   ```

   Choose A New Product, and press ⟨Return⟩.
7. The Packages menu appears. Select Entire to install all of the packages. You see the following two messages:

   ```
   Installing Custom Data Files...
   creating file lists...
   ```

8. You are prompted to insert distribution volume 1. Insert volume 1 of your distribution into the correct device driver, and press ⟨Return⟩.
9. If you selected Packages, a display of package names appears. To install all the packages at once, select ALL, and press ⟨Return⟩. If you wish to install more than one package, select the packages using the ⟨Space⟩ bar, and press ⟨Return⟩.
10. You see the following message:

    ```
    Extracting files...
    ```

    It can take several minutes for the files to be extracted from the volume.
11. You are prompted to insert distribution volume 2, if your distribution has more than one volume. Insert the next volume, and press ⟨Return⟩.

    Continue to insert volumes when prompted until all of the distribution files are extracted.
12. When all the selected packages of the distribution are extracted, **custom** runs the initialization script, if one is present in the permissions list. Then **custom** executes **fixperm**, which checks the file permissions, verifies that the files are installed in their correct locations, and creates directories and devices as specified by the permissions list. When this is done, the installation is complete.

    If you are finished with the installation, press any key to return to the **custom** menu. Select Quit and press ⟨Return⟩. Select Yes and press ⟨Return⟩ to exit.

Removing an application

After testing your distribution by installing it, you may now want to remove it from the system. This section describes the procedure for removing an application or packages from an application with the **custom** utility.

1. Log in to the system as *root*.
2. Type **custom** at the prompt, and press ⟨Return⟩.
3. Select Remove, and press ⟨Return⟩.
4. The Packages menu appears, along with a list of each package in the set. Select the appropriate package, and press ⟨Return⟩. To remove all the packages at once, select ALL, and press ⟨Return⟩.
5. At this point, **custom** runs any appropriate removal scripts, and removes the files listed in the permissions list. After a few messages asking you to verify the removal and telling you that the files are being removed, you return to the **custom** menu.

   Select Quit, and press ⟨Return⟩. Select Yes, and press ⟨Return⟩ to exit.

Software Mastering Toolkit utilities (SMT)

Intro

introduction to software mastering toolkit utilities

Description

This section contains manual pages for the utilities described in this guide. The following outline shows you how these tools fit in with the information presented earlier.

List of functions

Utility	Description
diskimage(SMT)	reformats data to fit a floppy disk
docut(SMT)	creates an application distribution
fdfit(SMT)	fits file archives onto media volumes
fdformats(SMT)	fits file archives onto floppies
hocheck(SMT)	compares a permlist with current and past distributions
mkcuts(SMT)	makes custom installable (application distribution cutting tool)
mkflops(SMT)	creates disks from disk images
mkperm(SMT)	makes a product permission list
permlint(SMT)	checks permlist syntax
pkgsize(SMT)	updates size information for package in special permlist comment
volno(SMT)	updates volume number information for files

diskimage

create file image for floppy disk

Syntax

diskimage ***disk_size***

Description

diskimage copies data from standard input to standard output, formatting so that the data is not larger than ***disk_size***.

docut

create an application distribution

Syntax

docut [-c] [-S] [-e*flag*] [-i] [-s] [-u] [-f *vars_file*] [-p *pkgdir*] [-o *mkperm_outfile*] [-O *mkperm_distdir*] [-D *mkperm_dirfile*]

Options

-c compresses the files in the distribution. **docut** also passes this option to **mkcuts.**

-C disables check for symbolic links

-e erases files specified by *flag*. *flag* is a combination of any of the following:

- **a** erases administrative files such as images and sums files
- **c** erases the CDIST directory, which contains compressed distribution files
- **d** erases the *dist* directory, which contains the uncompressed distribution files
- **p** erases the *misc* directory, which contains package descriptions previously entered (with the **-p** option) and starts from scratch

-f specifies a file that **docut** sources to set the required environment variables. If the **-f** option is not given, docut tries to source the file *./site_variables* by default.

-p cuts a distribution that is in multiple packages. It is included for backwards compatibility with the Product Engineering Toolkit (PET).

-S distribution contains files to be serialized

-o, -O, -D options that **docut** passes to **mkperm.** For details on these options, see the **mkperm**(SMT) manual page.

-i, -s options that **docut** passes to **mkcuts.** For details, see the **mkcuts**(SMT) manual page.

-u display command line usage, then quit

Description

The **docut** utility provides an interactive method of cutting an application distribution. **docut** first reads information about the product from the file specified with the **-f** option (or from the default file, *./site_variables*). It uses **mkperm**(SMT) to create a permlist for the product, and constructs a distribution tree containing all the files in the product. It then uses **mkcuts**(SMT) to transfer the distribution tree to the media, compressing the files if requested (with the **-c** option).

A simple use of **docut** is as follows: Copy all distribution files with the **tar**(C) utility into a new directory, for example, one named *./source*. (The *./misc*, *./dist* and *./cdist* directories are reserved names that cannot be used.)

Copy the file */usr/lib/scosmt/site_variables* into the current directory. Edit it to set the variables in it to the correct values for your product. If you rename this file, use the **-f** option to give **docut** its name. The following values are required.

PRODPRD	*prd* code for the product (*prd*= field in permlist)
PRODSET	full name of the product (*set*= field in permlist)
PRODTYP	system type of product (*typ*= field in permlist)
PRODREL	release number of product (*rel*= field in permlist)
BASEPACKDIR	directory under which the product's files are located, for example, *./source* above
BASEPACKNAME	name of the main package in the product
BASEPACKDESC	short description of the package
DEVICE	device name for archiving distribution
FORMAT	complete command needed to format a volume
VOLSIZE	size of each distribution volume
BLOCKING	blocking factor, if required

Execute **docut**. First, **docut** invokes **mkperm**. If the defaults are used, a permlist is created in *./dist/tmp/perms* and an image of your distribution hierarchy is created using links to the *./dist* directory.

Next, **docut** invokes **mkcuts** on the *./dist* hierarchy to allocate files to volumes and to transfer the product files to the distribution media. If you wish to inspect or edit the permlist, selecting Abort when prompted to insert the media will exit without writing to the media; selecting Continue will proceed to write out the product to the media.

If you selected Abort, either **docut** or **mkcuts** may be invoked again to complete cutting the product once you have finished making any changes to the permlist.

The **-c** flag is passed to mkcuts, which then creates a compressed hierarchy *./cdist* from which the distribution is cut. (Note that the *./dist* directory is still created and used as an intermediate step in this process.) The variable **NOCOMPRESS** in the variables file may be set to list packages that are not to have their files compressed. If **NOCOMPRESS** is blank or unset, the default action of the **-c** option is to compress all files in the product except the permlist.

Files

/usr/bin/mkperm
./misc/pkg.data
./misc/partperms
./misc/extrafiles

See also

fixperm(ADM), **mkcuts**(SMT), **mkperm**(SMT), **tar**(C)

fdfit

fits file archives onto media volumes

Syntax

fdfit [*options*] ... *size packages ...*

Options

-f *format* The files are saved in a *format* archive. The default *format* is the first listed in */etc/fdformats*, and by convention is **tar**(C). A number may be appended to *format*; the meaning of that number depends on *format* but usually indicates the blocking factor. For example, a *format* of **tar20** means that **tar** is blocked at 20.

The archive formats listed in */etc/fdformats* include:

tar[b] **tar** format, blocked **b**

cpio [c][B] **cpio** format, either binary or ASCII (**c**), and either default blocking or 5120-byte records (**B**)

ar **ar**(CP) archive

dd[*obs*] **dd**(C) format, with an output record size of *obs* bytes

-o *pattern* *pattern* is the naming convention used for the output files. Every time "#" appears in *pattern*, the two-digit number of the volume is substituted. There must be at least one "#" in *pattern*; the default is "#.fd". Volumes are numbered starting at one ("01"); if there are any files too big to fit on a single volume, these are listed as volume "00". Any old files following the naming *pattern* are removed.

-q does not complain about absolute pathnames, even if they are not recommended for *format*

-e exits successfully even if some of the files in the packages could not be found

-v prints, on the standard output, a table summarizing the arrangement chosen and the efficiency achieved

-F *formatsfile*
alternate list of *format* descriptions; the default is */etc/fdformats*

-O ***algorithm***
specifies the default arranging ***algorithm***:

n	Next file. If the next file in the input list does not fit on the current volume, the current volume is finished and a new volume started. This preserves the order of the files but may waste volumes.
f	First fit. The first file that fits on the current volume is chosen for that volume. Only when none of the files fit is the current volume finished and a new volume started.
b	Biggest fit. Similar to ***algorithm*** **f**, except that the largest file possible is chosen.
s	Smallest fit. Identical to ***algorithm*** **b**, except that the smallest file possible is chosen.
:	Each package starts on a new volume.
+	After one package is finished, the next package may be started on that volume.

Note that the default ***algorithm*** can be overridden on a package-by-package basis. The initial default ***algorithm*** is "f+".

-c checks to see if each package fits on exactly one volume. No output files are generated unless an explicit ***pattern*** is given, and all ***algorithms*** are ignored.

-l ***label*** The file named ***label*** (which is a pattern similar to **-o** ***pattern***) is the first file on each volume. If ***label*** does not exist, it is created with zero length and a mode of 0444 (only read access for all). If ***label*** does exist, it must be a regular file; whatever size ***label*** has is used.

size, which must be given after all ***options***, is the size of each volume in bytes. A suffix of "k" multiplies the number by 1024, "b" by 512, and "w" by 2.

packages are collections of files that are to be arranged as an indivisible unit. If the ":" ***algorithm*** modifier is in effect, each package starts a new volume; otherwise ("+") packages may share volumes. At least one ***package*** must be described after ***size***:

file The files in this package are listed in ***file***; the package has the default ***algorithm*** applied to it. A ***file*** of "-" means the standard input.

-p [***algorithm***] ***file***
Same as above, except that ***algorithm*** (if given) is used instead of the default.

-P [***algorithm***] ***files*** ...
Similar to **-p**, except that the named ***files*** are the package.

Description

fdfit tries to arrange groups ("packages") of files onto the fewest number of media volumes while avoiding splitting any file across two or more media volumes. Packages are not intermixed, and all links to a file within a package are placed on the same volume. Each volume has a capacity of ***size*** bytes, and is written in a specified ***format***, such as **tar**(C) or **cpio**(C).

An arbitrary number of packages may be given. The packages are dealt with on a one-by-one basis, with each package being completely arranged before the next package is started. The ***algorithm*** used to arrange each package may be individually specified. Whether or not a package can start on the same volume that holds the end of the previous package may also be specified.

The output of **fdfit** is a series of files; the first file lists the files that belong on volume one, the second lists the files for volume two, and so on (up to a limit of 100 volumes).

Notes

Links in packages arranged by the "n" algorithm are usually scrambled.

No effort is made to ensure that directories occur in the arrangement before any files contained in them.

Some formats are inherently non-portable. Furthermore, some of the portable formats, such as **tar**, have slightly different overheads on different machines.

Under obscure conditions, linked files in large packages may cause infinite looping.

Format file

An **fdformats** file describes a file archiving format to **fdfit**(SMT). **fdfit** only understands distributed dictionary archives, such as **tar**(C), **cpio**(C), and **ar**(CP).

Each line in an **fdformats** file describes a separate ***format***. The format is in the form:

fdformats ***volhdr filhdr volblk filmd*** [+] [***/***] [***type***[***size***]]...

Empty lines, and lines beginning with "*", are ignored.

The first field, ***format***, names the archive being described. ***format*** is an alphabetic name; no digits or non-letters are allowed.

The next four fields are decimal numbers giving the critical values:

volhdr the overhead for each volume (floppy). This is the sum of the sizes of the volume header and trailer blocks.

filhdr the overhead for each file (element) in the archive. This is the sum of the sizes of the element header and trailer blocks, and may be dependent on the length of the file's name. If so, a "+" should be specified in the miscellaneous fields section at the end of the line.

volblk the volume blocking factor. The number of bytes written to each volume is always a multiple of this size.

filrnd the file rounding boundary. Each element in the archive, with its header and trailer, is rounded up to the next such boundary.

Each of these values may have a suffixed unit of "w", "b", or "k", indicating multiplication of the number by 2, 512, and 1024, respectively.

In addition, each number may be prefixed by "#". This causes the number suffixed to *format*, as specified by the user with the **-f** *format* flag to **fdfit**, to be used instead of the value given in **fdformats**. If the value in **fdformats** is suffixed with a unit, the value from **-f** *format* is multiplied as indicated, ignoring any unit the user specified. But when **fdformats** does not indicate a unit, any unit indicated with **-f** *format* is used.

After the four values is a miscellaneous information section. If this contains a "+", then the length of a file's name must be added to that file's ***filhdr***. A "/" says that absolute pathnames are not a problem in this *format*.

The other fields in the miscellaneous section are a single character ***type*** optionally followed by a ***size*** value. The ***type*** indicates that a special file is correctly saved in a **format** archive:

b block special device

c character special device

d directory

n name space entry, such as semaphores and shared data regions

p named pipes (FIFO's)

& links to previously saved files

If ***size*** is a number (which may have a unit suffix), the ***type*** special file is always saved as if it were ***size*** bytes long. A ***size*** of "#" uses the actual file size obtained from **stat**(S). Otherwise, when no ***size*** is specified, zero (0) is assumed; the special file is completely described by the element header and trailer blocks.

Format file notes

Unpredictable results happen when both " + " and " & " are specified.

The meaning of " / " is subjective; most archiving programs archive absolute pathnames, but the behavior on extraction is undesirable (extreme measures like **chroot**(C) must be employed).

See also

ar(CP), **cpio**(C), **dd**(C), **tar**(C)

fdformats

fit file archives onto floppies

Syntax

fdformats ***format volhdr filhdr volblk filrnd*** [+] [/] [***type*** [***size***]]...

Description

An **fdformats** file describes a file archiving format to **fdfit**(SMT). **fdfit** only understands distributed dictionary archives, such as **tar**(C), **cpio**(C), and **ar**(CP).

Each line in an **fdformats** file describes a separate ***format***. Empty lines, and lines beginning with "*", are ignored.

The first field, ***format***, names the archive being described. ***format*** is an alphabetic name; no digits or non-letters are allowed.

The next four fields are decimal numbers giving the critical values:

volhdr overhead for each volume (floppy). This is the sum of the sizes of the volume header and trailer blocks.

filhdr overhead for each file (element) in the archive. This is the sum of the sizes of the element header and trailer blocks, and may be dependent on the length of the file's name. If so, a "+" should be specified in the miscellaneous fields section at the end of the line.

volblk volume blocking factor. The number of bytes written to each volume is always a multiple of this size.

filrnd file rounding boundary. Each element in the archive, with its header and trailer, is rounded up to the next such boundary.

Each of these values may have a suffixed unit of "w", "b", or "k", indicating multiplication of the number by 2, 512, and 1024, respectively.

In addition, each number may be prefixed by "#". This causes the number suffixed to ***format***, as specified by the user with the ***-fformat*** flag to **fdfit**, to be used instead of the value given in **fdformats**. If the value in **fdformats** is suffixed with a unit, the value from ***-fformat*** is multiplied as indicated, ignoring any unit the user specified. But when **fdformats** does not indicate a unit, any unit indicated with ***-fformat*** is used.

After the four values is a miscellaneous information section. If this contains a "+", then the length of a file's name must be added to that file's ***filhdr***. A "/" says that absolute pathnames are not a problem in this ***format***.

The other fields in the miscellaneous section are a single character ***type*** optionally followed by a ***size*** value. ***type*** indicates that a special file is correctly saved in a ***format*** archive:

- **b** block special device
- **c** character special device
- **d** directory
- **n** name space entry, such as semaphores and shared data regions
- **p** named pipes (FIFO's)
- **&** links to previously saved files

If ***size*** is a number (which may have a unit suffix), the ***type*** special file is always saved as if it were ***size*** bytes long. A ***size*** of "**#**" uses the actual file size obtained from **stat**(S). Otherwise, when no ***size*** is specified, zero (0) is assumed; the special file is completely described by the element header and trailer blocks.

Notes

Unpredictable results happen when both "**+**" and "**&**" are specified.

The meaning of "**/**" is subjective; most archiving programs archive absolute pathnames, but the behavior on extraction is undesirable (extreme measures like **chroot**(C) must be employed).

This is version 2.1 of the manual page, and describes the **fdformats** file expected by version 2.2 of **fdfit**.

See also

ar(CP), **cpio**(C), **dd**(C), **fdfit**(SMT), **stat**(S), **tar**(C)

hocheck

compare perms lists with current and past distributions

Syntax

/usr/bin/hocheck [-c] [-d] *destination_directory permlist* [*permlist(s)*] [-r *listing* [*listing(s)*]]

Description

hocheck takes the *permlist(s)* given to it and systematically goes through them, comparing the entries it finds with the files in the distribution tree. You must run **hocheck** from the root of the distribution tree you want to check. If told to check **cpio** or **tar** *listing(s)*, then **hocheck** also compares these lists with the permissions list(s) and the distribution tree. **hocheck** stores all the information it finds in a directory given by the user, *destination_directory*. It stores the following information in the following files:

devices lists all the devices given in the permlists, indicated by a *file type descriptor* of **p**, **b**, or **c**, and the major and minor device numbers that are given for the block and character devices. The description is preceded by the string *present* if the device has already been made in the distribution tree; otherwise it is preceded by the string *missing*.

distdirs lists all the directories that are in the distribution tree, but not on the permission list(s) or the **cpio**/**tar** listing(s)

distfiles lists all the files found in the distribution tree that are not on the permission list(s) or the **cpio**/**tar** listing(s)

empty lists all the empty files defined on the permission list(s)

links lists all links found in the permission list(s). If the links are actually just copies of each other in the distribution tree, then the list has the string *(second_file_a_copy)* in the first column. If the second file in the link is missing then the string in the first column is *(second_file_missing)*. Otherwise, the string *(normal_links)* is listed.

newdistfiles lists of files found in the distribution tree, but in neither the previous distribution tree (as per the last search by **hocheck**), nor the permission list(s) or **cpio**/**tar** list(s).

> **NOTE** This is only created when using the **-d** option.

packages lists of all the packages found in the permission list(s) and what permission list(s) they were encountered in

permdirs lists all the directories in permission list(s), but not distribution tree, regardless of whether they are listed in the **cpio/tar** list(s)

permfiles lists all the file found in the permission list(s), but not in the distribution tree, regardless of whether they are listed in the **cpio/tar** list(s)

ootdist lists all entries in the **cpio/tar** list(s) not found in the distribution tree

> **NOTE** This is only created when using the **-r** option.

ootperm lists all entries in the **cpio/tar** list(s) not found in the permission list(s)

> **NOTE** This is only created when using the **-r** option.

Options

-c Clean. This option causes **hocheck** to clean out the directory of previous files. If this option is not used, then all the previous files are backed up by appending them with a "**-**".

-d Differentiate. This option the creates the files ***newdistfiles***, which are brand new to this cut, and have somehow not been placed on the permission lists or **cpio/tar** lists. This is done by comparing the current ***distfiles*** with the previous ***distfiles***.

> **NOTE** This can only be done if **hocheck** has been run on a previous distribution tree and the ***destination_directory*** still exists, and is used again.

This option still works with the **-c** option.

-r Relative. This option signals **hocheck** that all following files listed on the command line are of the **cpio/tar** format of one field per line, each entry being a file or a directory listed relative to some *$ROOT*.

Files

/usr/bin/hocheck

Notes

If any file produced by **hocheck** is empty then it is not created.

See also

mkcuts(SMT)

mkcuts

make custom-installable distribution (application cutting tool)

Syntax

mkcuts [-c] [-i] [-s] [-u] [-f *vars_file*]

Options

- **-c** compresses the files in the product
- **-i** produces disk image files rather than cutting directly to the media
- **-s** produces a sumlist file listing the checksum of each file in the distribution
- **-f** sources the named file to set the required environment variables. If the **-f** option is not given, **mkcuts** sources the file *./site_variables* by default.
- **-u** display command line usage, then quit

Description

mkcuts is a tool for making **custom**-installable application distributions. **mkcuts** uses **fixperm** to extract the information in the permlist *./dist/tmp/perms/prd* to determine which files are on the distribution. Once it has this information, it runs a series of programs that create the distributions.

fdfit fits the files on the volumes in the most efficient order taking care to put specified files on specified volumes where indicated. For example, init scripts, in general, are on the last volume; installation tools are generally on the first volume. It creates files in **$MISCDIR**, which are named *01* ... *0n* and contain the list of files on that volume. **volno** looks in these files, **$MISCDIR/#**, and updates each file's entry in the permlist, *./dist/tmp/perms/prd*, with the corresponding volume number. **pkgsize** calculates the total size used by each package listed in the permlist and updates the permlist with this value. **fixperm** is run on the distribution hierarchy to set permissions and ownerships to match those of the permlist. You must be running as *root* so that **fixperm** runs correctly.

mkcuts uses a number of shell variables to obtain information about the product and the cutting environment. **mkcuts** sets these variables by sourcing the file given with the **-f** option, or the *./site_variables* file if no **-f** option is used.

The variables that need to be changed for each product cut with **mkcuts** are:

PRODSET full name of the product, for example, "SCO Professional"

PRODPRD short or abbreviated product name, for example, "pro"

PRODTYP code for the target machine and environment, for example "n386"

PRODREL current release number/name, for example, "3.2.0a"

PRODUPD update string; optional field only needed for an update, for example, "UA"

TARGET_OS operating system for which the product is intended; "UNIX" or "XENIX'

The variables that need to be changed for each different media type used in a cut are:

DEVICE device name for archiving distribution, for example, */dev/rfd096ds15*

VOLSIZE size of each distribution volume, for example, "1185k"

BLOCKING blocking factor (if required), for example, "10"

FORMAT complete command needed to format a volume, for example, **format -f** */dev/rfd096ds15*

NOTAR pathnames of any files not to be put onto the distribution media. This is used to stop link names from being put onto the media by **tar**. (Links are made by **fixperm** when the product is installed.)

Some special-purpose variables that are used for some products, but do not usually need to be changed from their default values, are:

SERIALKEY serialization key for serialization; optional, only if debranding files

SERIALIZE pathnames of any serialization files; optional, only if debranding files

VOLPREFIX optional prefix letters for the volume number field in the permlist.

NOCOMPRESS lists packages that are not to be compressed when the **-c** option is used. For example:

NOCOMPRESS="PERM INST TOOLS"

This ensures that the **PERM**, **INST**, and **TOOLS** packages in the product are not compressed when the other files in the product are compressed by **mkcuts**. If **NOCOMPRESS** is blank or not set, the **-c** option compresses all files in the distribution except the permlist.

The following variables define certain parts of the cutting hierarchy, and are almost never changed from their default values.

DISTDIR "root" directory of the distribution hierarchy, for example, *pwd/dist*

CDISTDIR "root" directory of the compressed distribution hierarchy, where the files in the distribution are compressed when the **-c** option is used

MISCDIR distribution "misc" directory, relative to **DISTDIR** used, for example, to store list of files corresponding to volumes

SUMDIR If a sum list of the final product is desired, **mkcuts** places it in the sum-directory listed, under the subdirectory **SUMDIR/PRODREL/MEDIA.**

IMAGEDIR directory under which the distribution image files are placed when the **-i** option is used. The files are placed under the subdirectory **IMAGEDIR/PRODREL/MEDIA.**

See also

fdfit(SMT), **fixperm**(M), **volno**(SMT)

mkflops

create floppy disks from mkcuts(SMT) output

Syntax

mkflops [-S] [-I *interleave*] [-d *devicetype*] [-o] [-p *prd*] [-r *release*] [-u]

Options

-I1 sets format interleave of 1

-I2 sets format interleave of 2

-d specifies the device for which the disk image is intended. *devicetype* must be one of the following: 48ds9, 96ds9, 96ds15, 135ds9, or 135ds18. **mkflops** does not request a device type later.

-o Other drive. It indicates that drive 1 is to be the output drive for cutting the distribution floppy disks.

-p specifies a product directory if the current working directory contains multiple product directories. **mkflops** does not request a product name later.

-r specifies a release number for the product if there are disk images for more than one release of the product. **mkflops** does not request a release number name later.

-S This option causes **mkflops** to ignore the sums left in the *KEY* sums file.

-u displays command line usage, then quits

Description

The **mkflops** utility takes disk images created by **mkcuts -i** and creates floppy disks using these images. Without any arguments, **mkflops** assumes that the image file or files are stored in the directory created by **mkcuts**, which has a pathname in this format:

IMAGEDIR/PRODREL/MEDIA

You can, however, run **mkflops** from a product directory or from the directory immediately above a product directory. This last option is particularly useful if you have a directory that contains several product directories because **mkflops** asks you to specify which product you wish to cut. Alternatively, you can specify a product with the **-p** option.

Notes

mkflops uses an interleave of two as standard formatting on disks.

Files

/usr/bin/mkflops

See also

mkcuts(SMT)

mkperm

make a product permissions list (permlist)

Syntax

mkperm [-p *prdvalue*] [-s "*setstring*"] [-t *type*] [-r *release*] [-S "*serialized files*"] [-o *output file*] [-O *dist directory*] [-x *volprefix*] [-D *dirfile*] [*pkgdir PKGNAME* "*pkgdesc*" [*pkgdir2 PKGNAME2* "*pkgdesc2*" ...]]

Description

mkperm creates a permlist from a directory hierarchy. This utility is called by **docut**(SMT). The resultant permlist is suitable as input for the **mkcuts** utility.

The following is a list of required values. If not specified on the command line, they are read from the appropriate environment variables. If a required value is not present on the command line or in the environment, **mkperm** gives an error message.

prdvalue: abbreviated product name; example: *pro*. Default value is **$PRODPRD**.

setstring: name of the full set or application; example: *SCO Professional*. Default value is **$PRODSET**.

type: target system. Default value is **$PRODTYP**.

release: release number of the product. Default value is **$PRODREL**.

pkgdir: root directory path for a given package

PKGNAME: name of the package associated with **pkgdir**

pkgdesc: description of the package

If *pkgdir*, *PKGNAME*, and *pkgdesc* are not given on the command line, they are taken from the variables **BASEPACKDIR**, **BASEPACKNAME**, and **BASEPACKDESC**.

The **-S** option sets the permlist variable **ser**, which lists the names of serialized files. If the **-S** option is not given, the **SERIALIZE** environment variable is used. If this is not set, a default value of "" (zero-length string) is used.

The **-x** option specifies prefix letters for the volume number field in the permlist. (This is mainly used for updates, and for the N, B, and X volumes of the OS.) If the **-x** option is not given, the value of the environment variable **VOLPREFIX** is used. If this is not set, no prefix letters are used.

If an output filename of "-" is given with the **-o** option, the permlist output is written to **stdout**. If the **-o** option is not given, the permlist output is written to **$DISTDIR/$PERMLIST** with a default value of *./dist/tmp/perms/prdvalue*.

The **-O** option specifies the root directory of the ***distribution tree***, which holds all the files to be copied onto the distribution media. **mkperm** creates this tree by making links to the appropriate files under the package directories. If a value of "-" is given with the **-O** option, this action is turned off; no links are made, and the distribution tree is not created. If the **-O** option is not given, the directory name is taken from the **DISTDIR** environment variable, with a default value of *./dist*

Note that the **fixperm -c** command must be run on the distribution tree before cutting the distribution media. This is necessary to set ownership and permissions, ensure that all links are made, and to create all directories and devices listed in the permlist. (This is done automatically if **mkcuts** is used.)

The **-D** option controls which directories are listed in the permlist. Without it, by default, **mkperm** omits from the created permlist any directories that are already listed in the operating system permlists (for example, *./etc*, *./usr*, *./usr/bin*). This is to ensure that custom-installable applications do not change the ownership/permissions on OS directories when installed. If this behavior is not appropriate (for example, when producing a permlist for an OS component) the **-D** option can specify a file that lists the directories to be omitted from the created permlist. The directories should be listed in the file one per line, without surrounding whitespace. (This is the same format as the output of **fixperm -D**.) Specifying a filename of "-" with the **-D** option makes **mkperm** include all directories in the created permlist, without omitting any.

Notes

mkperm must be run from the *root* login.

Any files whose uid or gid name cannot be determined are inherited by *root*.

Files

/etc/fixperm

See also

docut(SMT), **fixperm**(ADM), **fixperm**(M), **mkcuts**(SMT)

permlint

check permlist syntax

Syntax

permlint [-i] [-L] [-p] [-q] [-Q] [-u] ***permlist***

Description

permlint examines the permlist file, ***permlist***, and returns 0 if no syntax errors are encountered, or it generates a list of messages if it encounters errors or unusual permlist entries. Permlists are generated automatically by the **docut**(SMT) utility. Use **permlint** to check the syntax of permlists that have been edited manually.

Options

permlint supports the following options:

- **-i** suppresses error messages regarding UID/GID values that do not match those of the system
- **-L** suppresses warning messages regarding link names with volume numbers
- **-p** suppresses error messages regarding unusual permissions
- **-q** reports errors only
- **-Q** only reports syntax errors that **fixperm**(M) would detect
- **-u** displays command line usage, then quits

See also

docut(SMT), **fixperm**(M), **mkperm**(SMT)

pkgsize

updates package size information

Syntax

pkgsize [-cv] ***permlists***

Options

-c checks the files against the package size information in the ***permlist*** file, reporting any package sizes that are incorrect.

-v displays, for each package in the set (as indicated by the package name in the first field of the ***permlist***), a verbose listing of the number of blocks, files, links, directories, and devices. The final field listed indicates whether the package information was updated.

Description

pkgsize is designed for use with ***permlists*** of a special, extended format recognized by **custom**. Comment fields of a particular syntax, which are ignored by **fixperm**, contain the total size (in blocks) of each package within a set. **pkgsize** updates the numeric size information. For example, a ***permlist*** size comment might be:

```
#! PRO  1020    SCO Professional Spreadsheet Package
```

pkgsize would update the numeric field based on the total size of all files marked as being part of the PRO package.

Notice that you need to be sure that the files themselves are available in the same relative position as is indicated by the relative pathname specified in the ***permlist***. **pkgsize** recognizes link information, and only counts the size of linked files once.

See also

fixperm(ADM), **custom**(ADM)

volno

update volume number information for files

Syntax

volno [-ac] [-o *pattern*] [-f *types*] [-k *keyword*] *permlists*

Options

-a This option allows addition of volume number information if the volume number field was not yet included in the permlist. (The default action is to return an error during update if the volume number field is missing.)

-c This option does a check to see if any volume numbers need to be updated, and reports discrepancies without making changes.

-o *pattern* The *pattern* is matched for selection of file lists. Note that the *pattern* must contain a "#", which is then replaced with a filename corresponding to the two-digit volume number. For example, *dist/#* indicates that the file lists are in files named *dist/01*, *dist/02*, and so on.

-f *types* This option allows you to specify the types of files that occur on a distribution volume. This information is necessary because some archive programs do not include files of certain types.

The allowed file *types* are:

- **a** archive
- **b** block device
- **c** character device
- **d** directory
- **e** empty
- **f** regular
- **p** named pipe
- **x** executable

The default setting recognizes empty, regular, executable, and archive files.

-k *keyword* A *keyword* may be specified, which is then prepended by **volno** onto the volume number in the permlist. This is useful if there are several packages within a single distribution set, and the volumes within a single package are to be distinguished from those in other packages.

Description

volno is used with **fdfit**(SMT) to update a permlist suitable for use by **fixperm** with the appropriate volume number for each file in the package. **fdfit** produces a file for each volume needed, containing a list of the appropriate distribution files to be included in the named volume. **volno** then uses the information contained in these files to update a permlist file with the volume number for each file of a distribution.

The default *pattern* is that produced by **fdfit**.

See also

fdfit(SMT), **fixperm**(ADM)

Glossary

The following terms, used in the guide, are defined here for your convenience.

BINARY

A program consisting of executable code.

COMPATIBLE BINARY

A **BINARY** containing special information that allows it to be executed in a variety of environments.

CUT

The process of moving files from hard disk to removable media, such as floppy diskettes.

DISTRIBUTION

The physical media that contains the files necessary for installation of an application. Also used to refer to the files themselves.

FIXPERM

A utility that modifies files and devices to correspond to the information found in a permissions list.

IDD (Installable Device Driver)

Any product that links into the kernel.

INITIALIZATION SCRIPT

A shell script or program that the **custom**(ADM) utility executes at the end of a product installation; can provide special installation requirements.

INSTALLATION

The process of transferring files from removable media to a permanently accessible location (for example, a hard disk).

LABEL
A zero-length file, distributed on a media volume, which conveys a large amount of installation information via a series of carefully named nested directories.

PACKAGE
A logical division of an application that is presented by **custom**(ADM) as a separately installed entity.

PERM
A reserved package recognized by **custom**(ADM); the permlist and any preparation scripts are always included in the PERM package.

PERMLIST
A file that contains a list of all files, along with related information used by **fixperm**(ADM) and **custom**(ADM), necessary to install an application.

PREPARATION SCRIPT
A shell script or program that **custom**(ADM) executes before any other activity during installation; can be used, for example, to back up existing files before extracting new ones.

REMOVAL SCRIPT
A shell script or program that **custom**(ADM) executes before removing a package or product; can be used to provide special removal requirements.

SAMI (Semi-Automated Mass Installation)
An SCO product designed to aid in the process of installing a large number of systems by requiring only a single end-user-type installation of software.

SCRIPT
A series of Bourne shell commands gathered in a single executable text file designed to perform a task or series of tasks.

SET
All packages that together make up a product.

STANDARD INSTALLATION
A set of provisions that allows a product to be **custom**-installable and that conforms to suggested conventions.

USER-CONFIGURABLE
Any item that is specifically designed to allow users to make changes to meet their needs; also referred to as system-specific.

Index

H

I

L

M

N

O

P

R

S

T

U

V