NAME

mk – remake the binary system and commands from source code

DESCRIPTION

All source code for the UNIX system is distributed in the directory **/usr/src**. The directory tree rooted at **/usr/src** includes source code for the operating system, libraries, commands, miscellaneous data files necessary for the system and procedures to transform this source code into an executable system.

Within the **/usr/src** directory are the **cmd**, **lib**, **uts**, **head**, and **stand** directories, as well as commands to remake the parts of the system found under each of these sub-directories. These commands are named **:mk** and **:mk***dir* where *dir* is the name of the directory to be recreated. Each of these **:mk***dir* commands rebuilds all or part of the directory it is responsible for. The **:mk** command runs each of the other commands in order and thus recreates the whole system. The **:mk** command is distributed only to source code licensees.

Each command, with its associated directory, is described below.

:mklib The **lib** directory contains the source code for the system libraries. The most important of these is the C library. Each library is in its own subdirectory. If any arguments are specified on the **:mklib** command line then only the given libraries are rebuilt. The argument ***** causes it to rebuild all libraries found under the **lib** directory.

:mkhead The **head** directory contains the source code versions of the headers found in the **/usr/include** directory. The **:mkhead** command installs the headers given as arguments. The argument ***** causes it to install all headers.

:mkuts The **uts** directory contains the source code for the UNIX Operating System. The **:mkuts** command takes no arguments and invokes a series of makefiles that recreates the operating system.

Associated with the operating system is a set of headers that describe the user interface to the operating system. The source for these headers is found in a sub-directory within the **uts** directory tree. The user-accessible versions of these headers are found in the **/usr/include/sys** directory. The **:mksyshead** command installs these headers into the **/usr/include/sys** directory.

:mkstand The **stand** directory contains stand-alone commands and boot programs. The **:mkstand** command rebuilds and installs these programs. Note that these stand-alone programs are only applicable to the DEC processors and are not built for any other machine.

:mkcmd The **cmd** directory contains the source code for all the commands available on the system. There are two types of entries within the **cmd** directory: commands whose source code consists of only one file with one of the following suffixes: **.l**, **.y**, **.c**, **.s**, **.sh**, or a sub-directory that contains the multiple source files that comprise a particular command or subsystem.

Each sub-directory is assumed to have a makefile (see **make**(1S)) with the name *command***.mk** that takes care of creating everything associated with that directory and its sub-directories.

The **:mkcmd** command transforms source code into an executable command based on a set of predefined rules. If the **:mkcmd** command encounters a sub-directory within the **cmd** directory then it runs the makefile found in that sub-directory. If no makefile is found then an error is reported. For single-file commands, the predefined rules are dependent on the file's suffix. C programs (**.c**) are compiled by the C compiler and loaded stripped with shared text. Assembly language programs (**.s**) are assembled and loaded stripped. Yacc programs (**.y**) and lex programs (**.l**) are processed by **yacc**( ) and **lex**( ) respectively, before C compilation. Shell programs (**.sh**) are copied to create the command. Each of these operations leaves a command in the **./cmd** directory which is then installed into a user-accessible directory by using **/usr/sbin/install**.

The arguments to **:mkcmd** are either command names or subsystem names. Some subsystems distributed with the UNIX system are: **acct**, **graf**, **sgs**, **sccs**, and **text**. Prefacing the **:mkcmd** command with an assignment to the shell variable **$ARGS** causes the indicated components of the subsystem to be rebuilt.

For example, the entire **sccs** subsystem can be rebuilt by:

/usr/src/:mkcmd sccs

while the **delta** component of **sccs** can be rebuilt by:

ARGS="delta" /usr/src/:mkcmd sccs

The **log** command, which is a part of the **stat** package, which is itself a part of the **graf** package, can be rebuilt by:

ARGS="stat log" /usr/src/:mkcmd graf

The argument * causes all commands and subsystems to be rebuilt.

Makefiles throughout the system, and particularly in the **cmd** directory, have a standard format. In particular, **:mkcmd** depends on each makefile having target entries for **install** and **clobber**. The **install** target should cause everything over which the makefile has jurisdiction to be built and installed by **/usr/sbin/install**. The **clobber** target should cause a complete cleanup of all unnecessary files resulting from the previous invocation. The commands that use the **CLOBBER** environment variable are **:mkcmd**, **:mklib**, and **:mkuts**. These commands all check the **CLOBBER** variable before executing **make clobber**. If this variable is set to **OFF**, then **make clobber** is *not* performed. If the variable is *not* set or is set to anything other than **OFF**, the **make clobber** *is* performed.

An effort has been made to separate the creation of a command from source and its installation on the running system. The command **/usr/sbin/install** is used by **:mkcmd** and most makefiles to install commands in standard directories on the system. The use of **install** allows maximum flexibility in the administration of the system. The **install** command makes very few assumptions about where a command is located, who owns it, and what modes are in effect. All assumptions may be overridden on invocation of the command, or more permanently by redefining a few variables in **install**. The purpose of **install** is to install a new version of a command in the same place, with the same attributes as the prior version.

In addition, the use of a separate command to perform installation allows for the creation of test systems in other than standard places, easy movement of commands to balance load, and independent maintenance of makefiles.

SEE ALSO

install(1M), **make**(1S)