CHAPTER 12

SYSTEM UTILITIES

UNIX To Host File Copy Utility

12.1 INTRODUCTION

Since the Compiler, Timing Verifier, Simulator, and Packager run on a host computer, some method of tranferring files from the cluster to the host is needed. A simple file copy command (such as the one available in UNIX) is insufficient for several reasons. First, the files to be transferred reside in many different UNIX directories, but end up in one host directory making it possible to have file name conflicts. Second, there are many files in each UNIX directory that are not needed on the host (such as BODY drawings) and should not be copied. Third, only those files that have changed since the last time files were transferred to the host should be copied. Fourth, the UNIX file names are hidden from the user by the SCALD directories (which map SCALD drawing names to UNIX file names). The user should not have to understand this mapping just to transfer files when it need not be understood anywhere else. Fifth, the file transfer process should be automatic, transparent, and as easy to use as possible since the goal of the SCALDsystem is to make it easy to design logic and not to teach designers operating system details.

The SCALD file copy transfer utility satisfies these goals. It runs on the SCALDsystem cluster controller and transfers only those files needed for a compilation to the host.

12.2 WHAT DOES FILECOPY DO?

FILECOPY accepts a list of SCALD directories and other files that are to be transferred to a specific user on the host. The program decides which of the files specified has been changed since the last time FILECOPY was run and transfers only those files. The name of the file on the host will be the same as the name of the file in UNIX unless there is a name collision (two UNIX files with the same name) or the form of the name is incompatible with the host file name syntax (the name is too long, for example). The needs of the RSCS interface to the 370 and the DR-11W interface to the VAX are quite different and are described individually below.

FILECOPY WITH THE VAX

FILECOPY "knows" about the VAX interface and transfers files directly to the VAX via the DR-11W interface (which is called /dev/vms in UNIX). A large block size is used to make the transfer as fast as possible. When the HOST_KIND VMS directive is used, FILECOPY transfers files to the specified VMS destination (see below for a complete description of the use of the HOST_KIND directive). When FILECOPY completes, the files resides in the specified directory on the VAX.

FILECOPY WITH THE 370

The interface to the 370 is via RSCS. FILECOPY is used to create a list of files (as well as name translations) that are to be sent to the 370. Only those files that have been modified are listed. After FILECOPY has created this list (which appears in FILECOPY.LIST), the RSEND program is used to actually send the files to the 370. RSEND reads the FILECOPY.LIST file and generates commands that send each of the files to the 370 via RSCS. Once the files have been sent, the user must log onto the 370 and read the files from the reader. The GETRDR exec is used to read the files from the reader. After they have been read, the Compiler can be run. The HOST_KIND CMS directive causes FILECOPY to output a file list compatible with CMS. The OUTPUT_FILE_LIST ON directive should be used to cause the FILECOPY.LIST file to be created. These directives are discussed further below.

12.3 THE FILECOPY DIRECTIVES

The FILECOPY program reads directives from the filecopy.cmd file (in lower case). This file contains directives that specify the SCALD directories and files to be transferred, the name of the destination on the host, and the type of host. The filecopy.cmd file can be created with a text editor.

The directives understood by FILECOPY are described below. Each of them can appear as many times as desired. The directives names are NOT case sensitive, but the file names are (UNIX convention).

CMS_FILE_MODE

This directive is used to specify the file mode to be used when creating CMS format file names within SCALD directories on the host. This directive only applies when the HOST_KIND CMS directive is used (see below). The directive accepts a string. If the string is

empty, the filemode will be completely suppressed. If the string is not empty, the first character of the string is used as the filemode character. For example:

```
CMS_FILE_MODE '';       { output no filemode }
CMS_FILE_MODE 'c';      { use a filmode of c }
```

If no CMS_FILE_MODE directive is used, the Compiler assumes a filemode of 'a'.

COPY_FILES

This directive is used to specify a list of UNIX files that are to be transferred to the host. These files are not SCALD directories (SCALD directives are specified with the DIRECTORY directive as described below). The COPY_FILES directive is used to transfer individual files, such as the compiler.cmd file, that are not specified within SCALD directories (that is, they are not drawing files). Each UNIX file name must be unique since the UNIX path name is stripped off to create a file name on the HOST. FILECOPY does NOT change the name of these files when they are transferred to the host. This means that ALL of these files must be given names that are compatible with the host system. For CMS, names should be eight characters or less with an extension that is eight characters or less. For VMS, the name should be eight characters or less with an extension that is three characters or less. The directive accepts a list of file names (in quotes) separated by commas. There is no limit to the number of files that can be specified. All files must be text files (that is, files containing only ASCII data). An example of the COPY_FILES directive:

```
copy_files 'data', '/u0/george/complist';
```

If a COPY_FILES directive is not used, only the files specified in the SCALD directories are transferred.

DIRECTORY

This directive is used to specify the names of the SCALD directories to be transferred. The directive accepts a list of UNIX file names (the names of the directories). Each file name must be enclosed in quotes and separated by commas. There is no limit to the number of SCALD directories that can be specified.

FILECOPY does NOT change the name of the SCALD directory when it is transferred to the host. This means that ALL SCALD directories must be given names that are compatible with the host system. For CMS, names should be eight characters or less with an extension that is eight characters or less. For VMS, the name should be eight characters or less with an extension that is three characters or less. Some examples of the DIRECTORY directive:

```
DIRECTORY 'mydrawings.wrk';
directory '/u0/project/draw.wrk', 'drawings.wrk';
```

If a DIRECTORY directive is not used, FILECOPY only transfers those files specified with the COPY_FILES directive described previously.

HOST_DESTINATION

This directive is used to specify the destination directory name when transferring to the VAX. All files transferred by FILECOPY go to this directory; there is no provision for copying to multiple host directories. The form of the destination name is a path name with '/' separating path elements. The root of the path name is the name of the VMS interface device (/dev/vms). For example, the VAX directory [cpu.ibox.compile] would be specified as follows:

```
host_destination '/dev/vms/cpu/ibox/compile/';
```

Note that the path name MUST appear in quotes. Only one host destination is permitted. If no HOST_DESTINATION directive is used, FILECOPY prints an error message and copies NO files. The HOST_DESTINATION directive must ALWAYS appear. When transferring to the 370, a null HOST_DESTINATION should be specified as follows:

```
host_destination '';
```

HOST_KIND

This directive is used to specify the host system. The directive is specified as follows:

HOST_KIND VMS
: Used when transferring to the VAX. Causes FILECOPY to place a fully rooted VMS file name in the SCALD directory created on the

VAX. For example, files transferred to the VAX with HOST_DESTINATION '/dev/vms/mike'; will have fully rooted file names of the form [mike].

HOST_KIND CMS
: Used when transferring to the 370. Causes FILECOPY to create file names compatible with CMS in the SCALD directories created on the 370. Files transferred to the 370 are given names of the form filename.filetype.filemode. See also the CMS_FILE_MODE directive below.

HOST_KIND OTHER
: Used when no special file naming is desired. This is the default. All files in SCALD directories are given names of the form filename.filetype.

OUTPUT_FILE_LIST

This directive is used to cause FILECOPY to output a list of the files to be transferred rather than transferring them directly. This feature is always used when transferring files to the 370 since the file list is used by RSEND to send files via RSCS. The file list is also useful if transfer to tape is desired as is the case when using WALKNET. The file list is always written to the file FILECOPY.LIST (note upper case). The list of files specifies the same files as would be transferred if FILECOPY were to transfer directly to the host. Only those files that have been changed since the last run of FILECOPY (assuming the use of a transfer log file) are listed. The directive is specified as follows:

OUTPUT_FILE_LIST ON;
: Generate an output file list in FILECOPY.LIST and do not transfer the files directly.

OUTPUT_FILE_LIST OFF;
: Do not generate an output file list. Files are transferred directly to the host (use only if transferring to the VAX).

If no OUTPUT_FILE_LIST directive is used, FILECOPY transfers files directly to the host. This is an error if the host is a 370.

REPORT_FILES

This directive is used to control whether FILECOPY reports the UNIX file name and host file name as they are being copied. It has no effect when the OUTPUT_FILE_LIST ON directive has been specified (see above). The directive is specified as follows:

REPORT_FILES ON
: List each file to the terminal as it is being copied to the host. Has no effect if the OUTPUT_FILE_LIST ON directive has been specified.

REPORT_FILES OFF
: Do not report files are they are copied.

If no REPORT_FILES directive appears, the Compiler outputs no messages as the files are copied (default is REPORT_FILES OFF).

TRANSFER_LOG

This directive is used to specify the name of the file that contains a record of the last file transfer. This file is used by FILECOPY to determine which files have been changed since the last time the program was run. If errors were detected during execution, the transfer log is not updated. If the directives file has been changed since the last run of FILECOPY, the transfer log is ignored. The directive accepts a single UNIX file name enclosed in quotes. For example:

```
TRANSFER_LOG 'transfer.info'; {use transfer.info}
```

If no TRANSFER_LOG directive is used, no transfer log file is created, and FILECOPY always transfers all files specified.

THE FORM OF A DIRECTIVES FILE

The directives file has the following form:

```
<list of directives separated by semicolons>
END.
```

The "END." is used to mark the end of the file independent of the physical end-of-file. This is to make sure that some portion of the file has not been left off or that the file really is a directives file and not some random text file.

The semicolon separators are used to clearly mark the end of a directive and to give FILECOPY some error correction assistance when it finds a directive it cannot understand.

A directives file might appear as follows:

```
directory 'mywork.wrk';
directory 'drawings.wrk';
host_destination '/dev/vms/myroot/compile';
REPORT_FILES off;
copy_files 'compjob.com';
END.
```

Note that comments can be placed in the file if enclosed with '{' and '}'. Comments can appear anywhere a space can appear in the file. Line-feeds (end-of-line) can appear anywhere; directives need not all appear on one line. A file list for a single DIRECTORY directive can be placed on multiple lines if desired.

12.4 WHAT FILECOPY DOES

The following is a description of the steps performed by the FILECOPY program. It is only to help clarify the operation of the utility and should not be construed as a description of actual program flow.

1. Reads the directives file (FILECOPY.CMD). Any errors found are printed on the console.

2. Ensures that the files specified by COPY_FILES are unique, and transfers to the host (or outputs to the file list).

3. Read the SCALD directories specified. For each entry in each directory: does not transfer BODY or LOGIC files - only CONNECTIVITY files; does not transfer files that have not been changed; creates a unique host name for each file. Transfers each file to the specified directory on the host (or outputs to the file list). Reports the transfer if reporting is enabled (REPORT_FILE_LIST ON and OUTPUT_FILE_LIST OFF).

4. Transfers (oT outputs to the file list) the SCALD directory files themselves with file entries changed to reflect the names given the files when transferred to the host (which may be different than the UNIX file names because of name conflicts or file naming restrictions).

12.5 SPECIAL NOTES ABOUT FILECOPY

This section is included to provide some additional information about the FILECOPY program that might be helpful in interpreting its behavior.

LOCAL COPIES OF SCALD DIRECTORIES

The FILECOPY program will build SCALD directories on the host for the SCALD Compiler to use. The file names will NOT (in general) be the same as the file names used in UNIX. This is because the host file system is more restrictive than UNIX. FILECOPY creates a local copy of the SCALD directory with the new file names and saves it in the current UNIX directory (the directory in which the FILECOPY program is run). These SCALD directories are given names of the form:

XxX<name>

where <name> is the name of the SCALD directory. For example, the SCALD directory foo.wrk would be created in the local file XxXfoo.wrk.

FILECOPY does not remove these files when done. This is because the file may not have been transferred. If the OUTPUT_FILE_LIST ON directive is used, the files are to be transferred at some later time (by either WALKNET or RSEND) and the XxX files must remain until the transfer is preformed. Since the XxX files are not large, it is not expected that their existence will cause problems.

TRANSFER LOGS

FILECOPY keeps a record of the transfer if the TRANSFER_LOG directive is used. The transfer log file (specified with the TRANSFER_LOG directive) contains two items; a checksum of the FILECOPY directives file, and the system time of the last execution of the program.

The checksum is used to detect when the directives file has been altered. If altered, the transfer log is ignored and ALL files specified (with the COPY_FILES directive and within the SCALD directories) are transferred. This is necessary because FILECOPY does not keep a record of which files have been transferred; this information is implicit in the COPY_FILES and DIRECTORY specifications within the directives file. If the directives file changes, there is no longer any way of knowing which files have been transferred to the host since the set of files may have changed.

The second piece of information in the transfer log file is the system time of the last execution of the FILECOPY program. FILECOPY checks each of the SCALD directories specified. If it has not been modified since the last execution time (the time in the transfer log file), it is assumed that none of the drawings within the SCALD directory have changed. This is true since the Graphics Editor always modifies the SCALD directory whenever it write a drawing. For each drawing within a modified SCALD directory, FILECOPY checks to see if it has been modified since the last execution. If so, it transfers the file to the host. If any of the files within a SCALD directory are transferred to the host, the SCALD directory is also transferred.

OUTPUT FILE LISTS

When the OUTPUT_FILE_LIST ON directive is used, FILECOPY outputs a list of the files to be transferred rather than transferring them directly to the host. This file list has entries of the form:

'<drawing name>' <UNIX file> <host file>

where <drawing name> is the name of the SCALD drawing. If the file is not a drawing (as is the case with files specified with the COPY_FILES directive), the <drawing name> is the name of the file. <drawing name> is included to make it possible to output reasonable messages when processing the file list. <UNIX file> is the name of the UNIX file to be transferred to the host. The file has a path name that is correct for the current UNIX working directory but will only be fully rooted if it was specified as fully rooted in the FILECOPY directives file. <host file> is the name of host file to which the UNIX file is to be transferred. This name is either a fully rooted VMS file name (if HOST_KIND VMS directive was used) or a CMS file name (if HOST_KIND CMS directive was used).

The RSEND program reads this output file list and sends each of the specified UNIX files to the specified CMS file name via RSCS. The WALKNET program reads the output file list to produce a tape that can be read by either the 370 or the VAX.

12.6 EXAMPLE FILECOPY DIRECTIVES FILES

Examples are given below of the FILECOPY directives files for 370 and VAX file transfer. These are intended as a guide and should be modified to meet the user's specific needs.

DIRECTIVES FILE FOR THE VAX

Directives files for VAX transfers should always use the HOST_KIND VMS directive. This causes FILECOPY to create SCALD directories on the VAX with fully rooted VMS file names making it possible for any user to access drawings of another user by simply specifying the proper SCALD directory file name in the SCALD Compiler directives file. The Valid provided libraries can be accessed in the SCALD directories in [SCALD.LIBRARIES]. The OUTPUT_FILE_LIST ON directive should only be used when tape transfer is desired. A FILECOPY directives file may appear as follows:

```
directory 'user.wrk';          {the SCALD directory}
copy_files 'compiler.cmd',     {Compiler directives}
           'verifier.cmd',     {verifier directives}
           'packager.cmd';     {Packager directives}
transfer_log 'transfer.log'; {log file}
report_files on;               {report them as transferred}
host_kind VMS;                 {transfer to VMS}
end.
```

DIRECTIVES FILE FOR THE 370

The HOST_KIND CMS directive should always be used when using FILECOPY to transfer to a 370. The OUTPUT_FILE_LIST ON directive should always be used since FILECOPY does NOT know how to talk directly to the host. Once FILECOPY has been run, the user must ALWAYS run RSEND in order to transfer the files to RSCS and then must use the GETRDR exec to read the files from the reader on the 370. There is no way to determine when the transfer is complete other than waiting until there are no more ENQUEUED messages from RSCS. A FILECOPY directives file may appear as follows:

```
directory 'user.wrk';          {the SCALD directory}
copy_files 'compiler.cmd',     {Compiler directives}
           'verifier.cmd',     {verifier directives}
           'packager.cmd';     {Packager directives}
transfer_log 'transfer.log'; {log file}
host_kind CMS;                 {transfer to CMS}
output_file_list on;           {generate a file list}
end.
```

WALKNET Program

12.7 INTRODUCTION

WALKNET is a utility for transferring SCALD drawings and SCALD command files from the cluster to the HOST machine using half-inch, 1600 BPI magnetic tape. It is generally used in conjunction with the file copy utility.

12.8 GENERATING WALKNET TAPES ON THE CLUSTER

WALKNET takes a list of files, converts the files to the appropriate format and then writes them to tape. WALKNET takes two arguments:

WALKNET <host> <list of files> where

<host> is the host machine for which the tape is being generated. Legal values are -c, for the IBM/370 (CMS) and -u for the VAX (VMS) or the cluster (UNIX).

<list of files> is a list of the files to be transferred. This list is generated by the filecopy program.

To use WALKNET, complete these steps:

1. First run the filecopy program (details are found in the File Copy Utility description in this chapter and in FILECOPY HELP on the IBM/370 and FILECOPY.HLP on the VAX and Cluster). Be sure to include the OUTPUT_FILE_LIST ON directive in the filecopy.cmd file.

2. Mount a tape on the Cluster drive and place the drive on live.

3. Run WALKNET:

```
walknet -c FILECOPY.LIST
```

   (This example shows a tape being generated for an IBM/370, FILECOPY.LIST is the output file generated by the file copy utility.)

4. Dismount the tape and WALK to the the host machine.

12.9 GENERATING WALKNET TAPES ON THE HOST

This feature is not implemented yet.

12.10 READING WALKNET TAPES ON THE CLUSTER

This feature is not implemented yet.

12.11 READING WALKNET TAPES ON THE HOST

To read a WALKNET tape:

1. Logon to the host machine.

2. On an IBM system:

   a) Attach a tape drive, load the tape on the drive, and place the drive on-line.

   b) Be sure that you have the SCALD system software accessed. This can be done by:

      link scald 191 to <user> as <vaddr>
      access <vaddr> <unused filemode>

   c) Read the tape onto a disk by typing:

      walknet <file mode of disk you wish to copy to>

3. On a VAX system:

   a) Assign the tape drive, load the tape on the drive, and place the drive on-line.

   b) Read the tape onto a disk by typing:

      walknet <user directory you wish to copy to>

RS232 File Transfer Program

12.12 INTRODUCTION

The RS232 file transfer program uses a UNIX terminal board on the SCALDsystem and an RS232 port in the host system. Either a direct connection or a modem connection can be used.

12.13 BASIC PROCEDURE

In order to initialize the connection, you must run the file transfer program "cu" (/usr/bin/cu) on the SCALDsystem. This is done by typing in the following command:

```
cu /dev/ttyX [-t] [-s speed] [-l line] [- acu]
```

where speed should be replaced by the desired transmission speed and line should be replaced by the device name for the communications line device. Once the connection is established, you should log onto the remote system and initiate the com232 program. This program is responsible for turning off echoing and communicating with the file transfer program on the SCALDsystem.

Within "cu", lines beginning with ~ have special meaning. In order to transfer a file to the host system, you should use the following command:

```
(~%put) [fromfile] [tofile]
```

The command

```
(~%take) [fromfile] [tofile]
```

transfers files from the host to the SCALDsystem.

When the transfer is complete, you should log off the host.

Issuing the command ~. disconnects the line between the host and the SCALDsystem.

12.14 SYSTEM PREPARATION

1. Since com232 is a site dependent program, Valid supplies the source code (com232.pas). You must tailor it to your own environment and compile com232.pas.

2. You must also select a device /dev/ttyX from among the UTB devices (ttya - tth) to be dedicated to the RS232 file transfer link. The entry in /etc/ttys for this device must be set up to read 02ttyX.

Ethernet User's Guide

12.15 INTRODUCTION

The networking facilities of the 7.0 release of SCALDsystem have been totally redesigned. The facilities can be divided into two parts; the Extended File System or EFS which has been upgraded from the 6.0 release and the network facilities of BSD 4.1c UNIX which incorporate the remote login (rlogin), remote shell (rsh), rwho, and ruptime commands.

12.16 BASIC PROCEDURE

As part of the 7.0 installation procedure, the old passwd and group files are converted to new files by hashing the user name and group name. After the conversion, all users will have the new user id and group id for all files in the file system, and all nodes within the network will have consistency in user name, user id, and group id within the distributed environment.

NOTE

> Under no circumstance should the /etc/passwd and /etc/group files be changed. If there is any problem, contact VALID for assistance.

The booting procedure of the 7.0 UNIX system connects each node to the network through the "/etc/connioctl enable Hostname" command. This command initializes and sets up host table "/etc/hosts" for the network commands and then starts the network command daemons rshd, rlogind, and rwhod. When a user first logs onto the system, the system status of each node on the network can be checked through the following command:

```
/etc/connioctl show
```

EXTENDED FILE SYSTEM FACILITY

The 7.0 release of the Extended File System (EFS) allows read, write, close, and open operations to remote files using transparent file access. Under EFS, the system appears as a single machine with multiple file systems associated with remote machines. The files on the remote file systems are accessible through standardized UNIX naming conventions. The format of remote file pathnames is:

/net/<hostname>/<path relative to remote host>

For example, the file "/user1/wkfile1" on remote host "sys2" is accessed by the pathname "/net/sys2/user1/wkfile1." Similarly,

to "cat" a file "/tmp/foo" in the system "SCALD", type:

```
" cat /net/SCALD/tmp/foo "
```

to copy a file "/u0/class/sample" to the directory "/u0/test" on system "ENGR", type:

```
" cp /u0/class/sample /net/ENGR/u0/test "
```

to copy a file "/u0/class/sample" from the system "DEVELOP" to the current directory, type:

```
" cp /u0/class/sample . "
```

to copy a file "/etc/rc" on system "S1" to file /u0/temp/rc" on system "S2", type:

```
" cp /net/S1/etc/rc /net/S2/u0/temp/rc "
```

Note that it is illegal to touch more than one node in the pathname (i.e., "/net/foo/net/bar/tmp/foobar" is illegal).

EFS Operations

Release 7.0 provides a limited subset of operations to remote files (future SCALDsystem UNIX releases will support additional EFS operations). Currently, the following UNIX system calls are supported:

open(2)
close(2)
read(2)
write(2)
fstat(2)
stat(2)

In particular, it is now possible to open, close, read, and write a remote file. Attempting any other operation on a remote file that uses the EFS naming convention will result in an error. The following list represents a number of the UNIX utilities that can be used with EFS:

cat(1)
cc(1)
cmp(1)

cp(1)
diff(1)
ls(1)
more(1)
mv(1)
od(1)
tail(1)
vi(1)

Special files (block devices and character devices) should not be accessed by EFS calls although an error may not result. Namely, one cannot tar something to a remote tape drive. Future releases will support enhanced remote device manipulation.

File Security

EFS assumes that the password files (/etc/passwd and /etc/group) on the client and server machines contain the identical information. Specifically, every user on the local network or "site" must have the same user and group id numbers present in each machine on which they are granted file access. When making a remote call, EFS transmits the local user's user and group id numbers; the remote EFS server accepts the user id without question. To ensure a minimal level of security, each machine on the network should have an identical password file. When a new user is installed on any machine in the site by mkusr command, the user will be installed on every machine in the site. This network management facility has been implemented in this 7.0 release. In the 7.0 SCALDsystem UNIX release, the superuser on any machine in the site has superuser privileges on all machines at the site.

Some Notes On EFS

There are two types of users in the VALID network environment. The "mkusr" command determines which user has access the rest of the nodes in the network. If you are a local user only, you will not be allowed to access the network facilities at all. All the net-wide users are in the group named "net" in the /etc/group file.

The system manager is responsible for running "mkusr" to add the new user into the system. The new user will have the login directory and setup files created no matter in net or local mode. If in the net mode, the created new user is distributed over the net. Since the login directory and setup files are not created for the remote nodes in net mode, the system manager should take care of this by running the following command in other remote nodes if necessary:

```
mkusr -m username
```

When you use the "passwd" command to change the password, the new password is not distributed to other nodes. Accordingly, the password of a user might be different in different systems. The user should be aware of those changes for himself.

EFS is handled at the kernel level, and EFS transactions do not start shells in the remote machine as opposed to rlogin(1) or rsh(1).

The character special file "/net" (major/minor 18,0) is used as a hook by the kernel to enter the EFS layer. This file is installed by Valid. If "/net" is inadvertently lost or removed, catastrophic problems can occur especially in the initialization of the host table as described in the following paragraphs. The system manager should be able to fix this problem through the "mknod" command.

A machine cannot send EFS requests to other machines until this machine is enabled and connected to the network. Use the following command to see the status of all nodes on the network:

```
/etc/connioctl show
```

You may need to know the machine's name which you want to access beforehand (the above command will give you the necessary node information in the network).

Except for showing the status of the network, the connection manager in EFS allows the system manager to handle the network problems.

The /etc/connioctl has the following other arguments:

enable Hostname --- If the state is other than active when you do show, this command can restart the system.

disable --- Do this if you want to disconnect this node from the network for any reason.

listen --- Starts listening to the net.

nolisten --- Makes your system stop listening to the net.

shutdown period --- Broadcasts a "shutdown" message to the net.

See also UNIX Programmer's Manual --- open(2), close(2), read(2), write(2), stat(2), fstat(2), connioctl(8V), mkusr(8V), chknetusr(8V).

BSD 4.1C NETWORK FACILITIES

Remote Shell (rsh)

Rsh connects to the specified host and executes the specified command. Rsh copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Rsh normally terminates when the remote command does.

The remote uid and gid used are the same as your local uid and gid. The provision is made for specifying the username or password with the command.

Shell metacharacters (< , > , >> , | , { , } , [,] , ;) which are not quoted are interpreted on local machine, while quoted metacharacters are interpreted on the remote machine. Thus, the command:

```
rsh otherhost cat remotefile >> localfile
```

appends the remote file remotefile to the local file localfile, while:

```
rsh otherhost cat remotefile ">>" otherremotefile
```

appends remotefile to otherremotefile.

Host names are given in the file /etc/hosts. This table is automatically maintained by the /etc/chkhosts network management command and will be consistent with the incore host table shown by the connioctl command.

After rsh makes the connection with remote host, you are at the root directory. Also your local uid and gid are used in the remote host for checking the access privileges.

You cannot run an interactive command (e.g., mail, vi, ex, ed) or interactive shell script or user program.

Stop signals stop the local rsh process only.

See also /etc/hosts, rsh(1), inithosts(8V), chkhosts(8V).

Remote Login (rlogin)

Rlogin connects your terminal on the current local host to the remote host system. When you rlogin as the same user on an equivalent host, you don't need to give the password. If the originating user is not equivalent to the remote user, then a login and password will be prompted for on the remote machine as in login(1).

Your remote terminal type is the same as your local terminal type (as given in your environment TERM variable). All echoing takes place at the remote site so that (except for delay) the rlogin is transparent. Flow control via CTRL-S and CTRL-Q and the flushing of input and output on interrupt are handled properly. A line of the form "~." disconnects from the remote host. Also you will lose connection by typing CTRL-D.

Refer to the rlogin(1) in UNIX Programmer's Manual for details.

Rwho and Ruptime

The rwho command produces output similar to who, but for all machines in the local network.

The ruptime command gives the host status for each machine on the local network. It prints the current status of up or down, the length of time the system has been up, the number of users logged into the system, and the average number of jobs in the run queue over the last 1, 5, and 15 minutes.

See rwho(1) and ruptime(1) in the UNIX Programmer's Manual for details.

Some Notes On Network Command

A machine cannot execute network commands on other machines until the command daemon is running and all the machine names are included in the /etc/hosts host table. Since the /etc/hosts table is maintained dynamically, you should cat /etc/hosts file to see if the desired host name is in the table in case there is difficulty in accessing the network. Also the following command will give you all the status about the nodes in the network:

```
/etc/connioctl show
```

You may need to know the machine's name which you want to access beforehand (the above command will give you the necessary node information in the network).

In case the /etc/hosts host table is damaged, the system manager should be able to fix it by the following commands:

```
/etc/inithosts Hostname
/etc/chkhosts
```

These commands initialize the host table and add the up-running machine name on the network into the /etc/hosts table.

See Also UNIX Programmer's Manual --- rsh(1), rlogin(1), rwho(1), ruptime(1), rshd(8), rlogind(8), rwhod(8), connioctl(8V), mkusr(8V), chknetusr(8V), iniithosts(8V), chkhosts(8V).