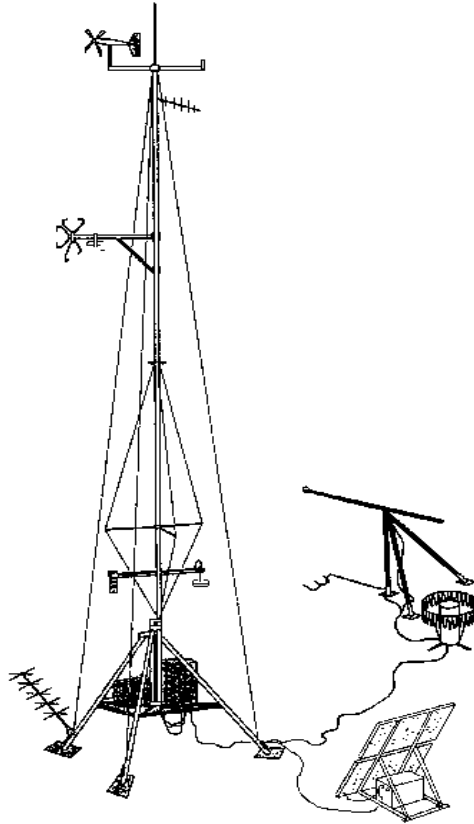


# PAM-III Base System Software Guide

(Preliminary Draft, Sept 1997)



National Center for Atmospheric Research  
Surface and Sounding Systems Facility  
Boulder, CO 80307-3000

# PAM-III Base System Software Guide

1.0	Directory Structure.....	1
2.0	UNIX User Information.....	1
2.1	Login Sequence.....	1
2.1.1	Environment Variables.....	1
3.0	System Management.....	3
3.1	Building Pam Software.....	3
3.1.1	Complete Software Build.....	3
3.1.2	Partial Build.....	3
3.1.3	Compiler Warnings and Errors.....	3
3.1.4	Splus Errors.....	4
3.1.5	Architecture Mismatch Errors.....	4
3.1.6	Build Permission Errors.....	4
3.2	Syslog.....	4
3.3	Pam Group.....	5
3.4	Boot-up Script.....	5
4.0	Maintaining EVE Configuration.....	6
4.1	Direct Download of Configuration from Base to EVE.....	6
4.2	Direct Upload from EVE back to Base.....	7
4.3	Transferring an EVE Configuration from Base to PC.....	8
4.3.1	From Base To PC Via Floppy.....	8
4.3.2	From Base To PC Via Network.....	9
4.4	Downloading Configuration from PC to EVE.....	9
4.4.1	From PC to EVE via Flash Card.....	9
4.4.2	From PC to EVE via Ymodem.....	10
5.0	Base System Programs and Scripts.....	10
5.1	Packet Decode (pdecode).....	10
5.2	Packet Archive (parch).....	13
5.3	Poll Pam Stations (ppoll).....	14
5.4	polln.....	17
5.5	eve_console.....	17
5.6	flash.....	18
5.7	logbook.....	18
5.8	ncprint.....	18
5.9	utime.....	19
5.10	Sun_angle.....	19
5.11	Satellite_angle.....	19
6.0	Assorted UNIX Information.....	20
6.1	Editing Files.....	20
6.1.1	dtpad editor.....	20
6.1.2	The joys of vi.....	20
6.1.3	Help I'm in ed!.....	21
6.1.4	emacs.....	21

## 1.0 Directory Structure

The PAM base software and configuration information is organized in one directory tree. This directory tree can be placed anywhere within the system directory structure. See Figure 5, “PAM Directory Structure,” on page 5. Before a process can use the PAM software, the path to the PAM directory must be placed in a UNIX environment variable, called **PAM**. The PAM software uses the value of this variable when creating directory paths for configuration and data files. The **PAM** environment variable can be defined automatically for users during the login process

## 2.0 UNIX User Information

### 2.1 Login Sequence

The PAM software can be run from any login account on the system, as long as it has permission to read, write and execute files on the PAM directory. It is recommended that a pam group be created, and all files on the PAM directory be owned by the pam group.

In order to set up the environment for the user upon login, add the following lines to the user’s .login file:

```
setenv PAM /pam_root_directory
setenv PROJECT default_project          # optionally set default project
source $PAM/scripts/pam.login
```

and to their .cshrc file:

```
setenv PAM /pam_root_directory
source $PAM/scripts/pam.cshrc
```

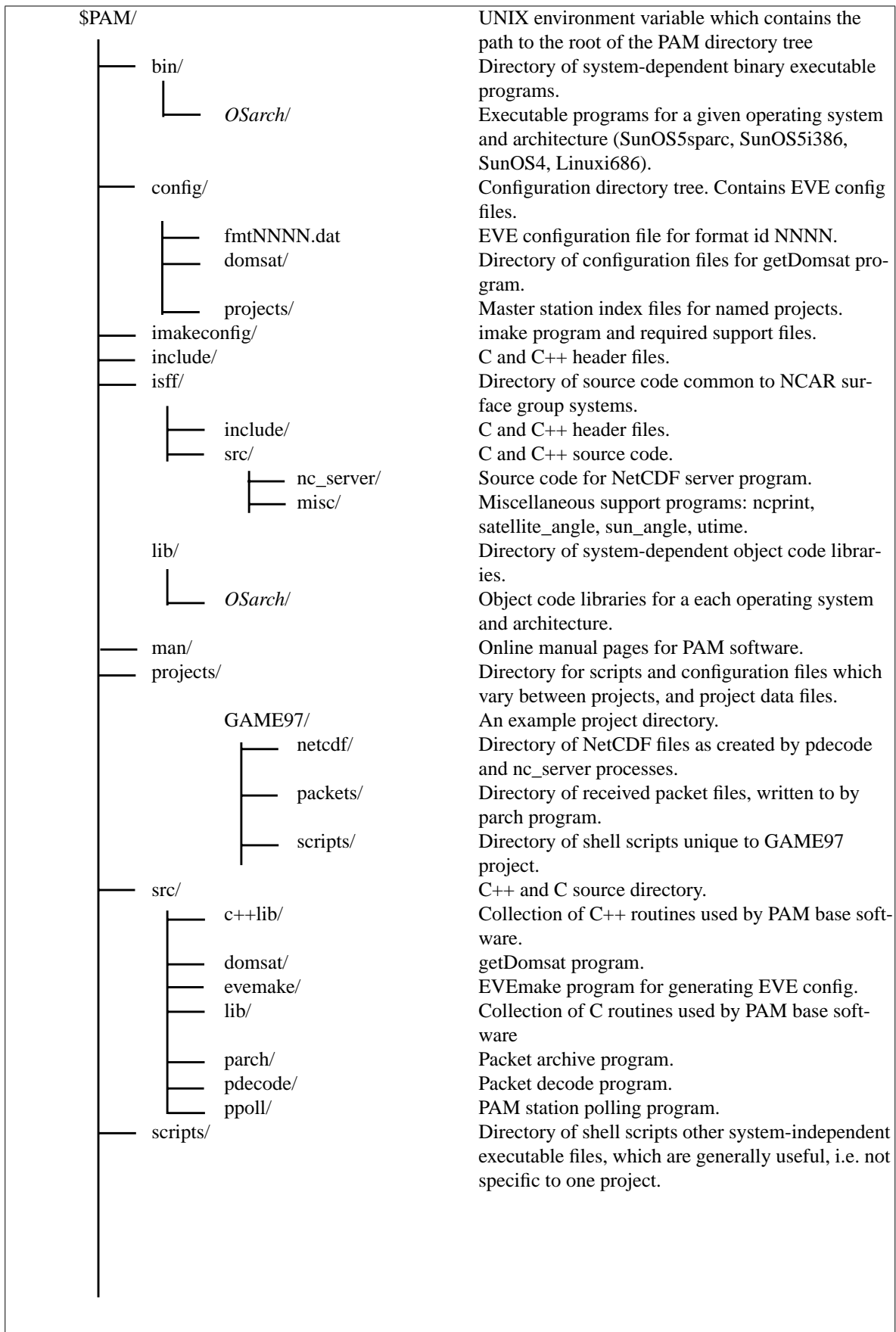
*/pam\_root\_directory* is the name of the PAM directory. *default\_project* is the name of the PAM project that the user is currently working on (which should be the name of a directory in \$PAM/projects). If the **PROJECT** environment variable is not set in the .login file, as shown above, the **pam.login** script looks for a file called **\$HOME/default\_project** in the user’s home directory. If that file exists, it should contain one word on one line, the name of the default project. If the **PROJECT** environment variable is not set via either of these ways, the user will be prompted to select a project. When logging in to a X window system which uses a display manager, or when starting PAM processes at bootup, it is usually not possible to prompt the user for input. In this case, it is recommended that the *default\_project* be set for the user in one of the two ways above. A **pam\_project** command is provided to allow users to switch projects after login.

#### 2.1.1 Environment Variables

In addition to **PAM** and **PROJECT**, the **pam.login** script also customizes the **PATH**, **MANPATH**, **SURF\_NETCDF\_DIR**, **SURF\_NETCDF\_FILE**, **EVE\_CONFIG** and **PAM\_NETCDF** environment variables.

The following directories are appended to **PATH**, in order:

```
$PAM/projects/$PROJECT/scripts
```



**FIGURE 1. PAM Directory Structure**

```
$PAM/scripts  
$PAM/bin/OSarch
```

This defines the search sequence that the UNIX shell uses when looking for an executable program or script. Therefore, if a project needs a special version of a script, it can be placed on the project scripts directory, and it will be invoked instead of a version on \$PAM/scripts.

The location of the PAM manual pages (**\$PAM/man**) is appended to **MANPATH**, so that they can be accessed online, with the **man** command.

The other environment variables are specific to PAM, and are discussed under the PAM Programs section.

## 3.0 System Management

### 3.1 Building Pam Software

#### 3.1.1 Complete Software Build

To do a complete build of the PAM software, login as a pam user, and do:

```
cd $PAM  
make -f Makefile.ini Makefile  
make World >&! World.Log  
more World.Log          # Look for errors listed below  
make install >&! install.Log  
more install.log
```

#### 3.1.2 Partial Build

To do a partial build in a specific directory, for example \$PAM/src/ppoll, do:

```
cd $PAM/src/ppoll  
pmkmf  
make depend  
make clean  
make install
```

The “pmkmf; make depend; make clean” steps are not always necessary, but are wise to do if any major changes have been made to the code, or if one is doing builds in the same directory from systems of different architectures, see Section 4.1.5 Architecture Mismatch Errors.

#### 3.1.3 Compiler Warnings and Errors

Most compiler warnings are benign, including the following:

```
warning: return makes pointer from integer without a cast  
warning: assignment makes pointer from integer without a cast  
warning: implicit declaration of function `int gethostname(...)'  
warning: passing arg 2 of `signal' from incompatible pointer type
```

```
cc1plus: warning: -g option not supported for C++ on systems using the DWARF debugging format
```

Compiler errors, however are usually serious. Please report any errors to PAM Software Technical Support, PAM International Plaza, Boulder CO (1 800 GOODLUCK).

### 3.1.4 Splus Errors

Splus may not exist on every system. However, "make World" will still try to build the Splus libraries, and you may get these errors, which can be ignored.

```
Splus.compile -D /home/pam/.Data dat.q && touch .fun.dat
Splus: Command not found
*** Error code 1 (ignored)
```

### 3.1.5 Architecture Mismatch Errors

These errors are not good and must be corrected:

```
ld: fatal: file xxxx.o: wrong machine type
ld: fatal: File processing errors. No output written to yyyy
ld: fatal: file /home/pam/lib/SunOS5sparc/libpam_c++.a(pam.o): wrong machine type
ld: fatal: File processing errors. No output written to parch
```

They occur if systems of different architecture (Intel X86 and SPARC) are sharing the same PAM directory tree, and the loader encounters object files or libraries of the wrong machine type. If they occur, it is recommended that you do complete software builds on each system. Don't do the builds simultaneously. Wait until one build is complete and installed with no errors before doing the build on the second system. If you understand the make structure, you can do a partial build of the effected files, making sure you do the "pmkmf; make depend; make clean" steps.

### 3.1.6 Build Permission Errors

These installation errors occur because file ownership is not set correctly. The files are otherwise installed correctly, and the errors can be ignored. Or they should be simple to correct by changing ownership with the chown command.

```
/usr/sbin/install -f /home/pam/lib/SunOS5sparc -m 0664 libnetcdf_c++.a
chgrp: /home/pam/lib/SunOS5sparc/libnetcdf_c++.a: Not owner
chown: /home/pam/lib/SunOS5sparc/libnetcdf_c++.a: Not owner
chmod: WARNING: can't change /home/pam/lib/SunOS5sparc/libnetcdf_c++.a
libnetcdf_c++.a installed as /home/pam/lib/SunOS5sparc/libnetcdf_c++.a
```

## 3.2 Syslog

The PAM base programs can optionally (with **-S** option) send error and information messages to the system log process, **syslogd**. When using syslog, the messages are merged in time sequence in the same file, each with a time stamp, process name and process id. The configuration file `/etc/syslog.conf` controls the disposition of the syslog messages. Adding the following lines to `/etc/sys-`

log.conf will cause all PAM syslog messages to appear on the console, and messages of severity “notice” or higher to be written to a disk file.

```
# PAM syslog messages
local0.debug           /dev/console
local0.notice          /var/log/local/pam.log
```

Note: the white space between the two fields must be tabs, not space characters. This file can only be edited by the super user. The actual disk path to the log file must be used, the PAM environment variable is not available. One other gotcha is that the log files must be on a disk that is mounted to the system early in the boot sequence; don’t put the log file on a disk that is auto-mounted. If you’re in doubt, put it on /var/log/local. The log file must exist. To create the initial file, enter:

```
mkdir -p /var/log/local
touch /var/log/local/pam.log
```

Once /etc/syslog.conf has been edited, and the log file created, you must reboot or send a hangup signal to the syslogd process for the changes to take effect:

```
kill -HUP `cat /etc/syslog.pid`
```

### 3.3 Pam Group

A convenient way to allow access for multiple users to the PAM directory is to create a pam group. Files in the \$PAM directory should be owned by this group, and users of the PAM software should belong to the pam group. The pam group can be created with admintool or by editing /etc/group from the super-user account.

### 3.4 Boot-up Script

If it is desired to automatically start up PAM base processes when the UNIX system is booted, one can create boot scripts to do so. Under Solaris 2.x, create a script called /etc/rc3.d/S80pam, containing the following:

```
#
# Start PAM processes
#
user=pam
case "$1" in
'start')
    echo "Logging in as $user to start PAM processes..." >/dev/console
    su - $user -c "source .login; pam_start" >/dev/console 2>&1
    echo "PAM processes started" >/dev/console
    ;;
'stop')
    su - $user -c "source .login; pam_stop" >/dev/console 2>&1
    ;;
esac
```

Choose the user (in this example, pam) under whose account you want to run the PAM processes. This example will run the scripts **pam\_start** on bootup, and **pam\_stop** on shutdown. These scripts should exist either in **\$PAM/projects/\$PROJECT/scripts** or **\$PAM/scripts**.

## 4.0 Maintaining EVE Configuration

The EVE configuration files are kept in the **\$PAM/config** directory, with names like **fmtNNNN.dat**, where **NNNN** is the four digit format number. See the [EVE Users Manual](#) for information on the format of the configuration files. In order for the packet decoding process to work correctly on the base system, the configuration running on each EVE processor must be identical to the corresponding configuration file on **\$PAM/config**. Also, in order to keep track of the current EVE configuration, any changes to an EVE configuration file should be made on the base system, and then transferred to a PC for download to EVE.

If a direct serial link exists from the base to a station (as over an RF modem), configuration files can be directly downloaded from the base. Otherwise, the process usually involves transferring the file to a notebook PC, either via a floppy disk or over a network connection, and then downloading the file from the PC to EVE at the station.

### 4.1 Direct Download of Configuration from Base to EVE

This example will show how to transfer an EVE configuration number 54 to station 2, from the base system, over an RF modem serial link. The transfer uses the **xmodem -y** (YMODEM) command on the EVE side, and the **sz** (X-Y-ZMODEM) command on the UNIX side from the **tip** program, which is run by **eve\_console**. Text files must be transferred in binary mode (**-b** option), otherwise extraneous CR characters are added. The trick is to use “~C” (tilde,C) to escape out of tip in order to execute the two commands.

From the system directly connected to the RF modem (at SHEBA, that is pampoll), do the following. User entries are in **bold**, informative comments are after the # sign:

```
cd /tmp
evemake 54           # creates a sNN_0054.dat file for each station NN.
                    # Don't use -d (DOS) option.
wc s02_0054.dat     # note the size in bytes (third number shown) of the UNIX file.
eve_console        # shuts down polling, runs tip
console002         # get station 2's attention. If necessary, keep typing this
                    # until you get the Eve> prompt.

Eve> entersys
System EVE# dir
                    # make sure there are enough "Bytes Free" for new file.
                    # If not, clean out (with del comand) any obvious junk.

System EVE# xmodem -rby s02_0054.dat
XMODEM Version 3.10.vxW.02 (20 Apr. 1994) -- vxWorks File Transfer Facility
Ready to RECEIVE File S04_0054.DAT in binary mode.
Send several Control-X characters to cancel
~C                   # type ~C (tilde, then capital C). You should receive the
                    # Local command? prompt. Try again if necessary.
~CLocal command? sz s02_0054.dat
```

```
Retry 0: Awaiting pathname nak for s02_0054.dat
Sector 161 20k Retry 0: Awaiting pathname nak for <END>
Sector NN Nk      # numbers should increment
away for NN seconds
!
```

The sector and kbyte counts should increment, indicating the transfer is working. Be patient. If it isn't happening, hit return. If all else fails, type Control-C (hold down Control key, press C), then Control-X until you get the System EVE# prompt, and then try again.

```
System EVE# dir      # Check that s02_0054.dat file exists, and is the same size as on
                    # the UNIX side. If it is the same size, all should be OK. If not, you
                    # need to re-transfer.
```

If you want to double check that the file transferred correctly, you can transfer it back to UNIX and check that the file is identical to the original. See Section 5.2 Direct Upload from EVE back to Base.

If the file on EVE looks OK, make a backup of config.dat as config.old, copy s02\_0054.dat to config.dat, and restart EVE:

```
System EVE# copy config.dat config.old
System EVE# copy s02_0054.dat config.dat
System EVE# start  # Restarts the EVE process.
```

At this point EVE will disconnect from the AUX Console port, and the EVE system console port (the one you connect to when visiting the station) becomes active. You must wait 5 minutes for this port to timeout and become inactive before you can access the AUX console port again.

```
console002
System console in use      # wait five minutes
console002
Eve> status              # browse around to see that all is OK
Eve> exit
~.                          # tilde, period to exit tip
```

If you can't get the station's attention, try again in 5 or 10 minutes. If it still doesn't respond, then it's probably hung and must be rebooted. Heck, after all this tedious conversation with dim-witted computers, it's time to stretch your legs anyway.

## 4.2 Direct Upload from EVE back to Base

Assuming you are running **eve\_console**, and have the "System EVE#" prompt from a station, do the following to copy a file from EVE to the Base system. For some unknown reason YMODEM doesn't seem to work, so you must use the EVE **xmodem** command without the **-y** option, and do the transfer in ASCII, not binary mode. This example copies the file s02\_0054.dat from EVE to a file called test.dat on the Base:

```
System EVE# dir s02_0054.dat    # check size of file
System EVE# xmodem -st s02_0054.dat
File ./S02_0054.DAT Ready to SEND in text mode
```

```

Estimated File Size 21K, 161 Sectors, 20538 Bytes
Estimated transmission time 24 seconds
Send several Control-X characters to cancel
~C          # type ~C (tilde, then capital C). You should receive the
            # Local command? prompt. Try again if necessary.
~CLocal command? rx -a test.dat
Incoming: 2000000000 0 0
Receiving ASCII w
rz: ready to receive test.dat
001 002 003 004 005 006 007 008 009 010      # numbers should increment
Please read the manual page BUGS chapter!    # ignore it
away for 42 seconds
!
System EVE#

```

After the transfer completes, check the size of test.dat from another UNIX window:

```

cd /tmp
wc test.dat      # the file size is the third number

```

It may be smaller than s02\_0054.dat on EVE, because this transfer seems to remove trailing blanks in each line, which is OK. If this upload was done to double check that a download was successful, you can check that the files are otherwise identical with the UNIX **diff -w** command:

```

diff -w s02_0054.dat test.dat

```

If **diff** just returns without reporting anything, the files are identical, except for possible trailing white space on some lines. Otherwise it will report conflicting lines.

Back in the **eve\_console** window, finish your EVE session, and exit **tip** with ~. (tilde,period).

### 4.3 Transferring an EVE Configuration from Base to PC

First, configuration files must be made for each station, using the **evemake** script, as shown below, and transferred to the PC. This can be done via floppy disk or over the network, if available.

The **evemake** script uses the value of the **PROJECT** environment variable when opening the file **\$PAM/config/projects/\$PROJECT**. This file contains a list of the active stations. **evemake** will make a configuration file for each of the active stations.

#### 4.3.1 From Base To PC Via Floppy

Insert a formatted floppy disk into the base system floppy drive.

```

volcheck
cd /floppy/floppy0
evemake -d N

```

Where N is the format number, for example, 51.

The **evemake -d** option causes the files to be converted from UNIX ASCII file format to DOS file format. Whether or not to use this option depends on how the file will be transferred from the PC to EVE. If the file will be transferred via flash disk, then do not use **-d**. If the file will be transferred via Xmodem/Ymodem via the EVE console port, then use **-d**.

Evemake will create a format file for each active station, with names like sXX\_NNNN.dat, where XX is the two digit station number, and NNNN is the four digit format number.

Eject the floppy:

```
cd
eject
```

Insert the floppy into the PC, and copy the files from the A: drive to the appropriate project directory on c:\PAM, for example c:\PAM\SHEBA.

### 4.3.2 From Base To PC Via Network

This assumes the network connection between the notebook PC and the base system is operational, i.e. the base system **\$PAM** directory is accessible as a network disk from Windows95 on the PC. Other PAM system documentation describes setting up the network connection and configuring SAMBA.

On the base system:

```
cd $PAM/config
evemake -d N
```

See above about the evemake -d option.

On the PC, drag and drop the files from the PAM network disk to the appropriate project directory on c:\PAM, for example c:\PAM\SHEBA.

## 4.4 Downloading Configuration from PC to EVE

### 4.4.1 From PC to EVE via Flash Card

If downloading the configuration via flash card, the file should have been created by evemake, without the **-d** option.

After inserting a flash card into the PC, it should appear as the D: or E: drive.

If the card is new, or EVE has had difficulties writing to it, or if you just want to clear files from it, you can format it with the following steps:

1. Double click on the "My Computer" icon.
2. Single click with the right button on the flash disk icon, which will be either the D: or E: drive.
3. Select "Format"

Then copy the files by drag and drop from the c:\PAM project directory to the flash card.

Take this card to the EVE system. Connect the PC to the EVE console with a serial cable and run ProComm on the PC.

Hit return if necessary in the ProComm terminal window to get the EVE prompt, then shut down EVE processing with the stop EVE command:

```
Eve> entersys  
System EVE# stop
```

Insert the flash card, then copy the file to EVE's ramdisk, and then to config.dat:

```
System EVE# copy /card/sXX_NNNN.dat sXX_NNNN.dat  
System EVE# copy sXX_NNNN.dat config.dat
```

where XX is the station number, and NNNN is the format number. Make sure you use the correct station number.

Restart EVE:

```
System EVE# start
```

Watch the console for errors.

#### **4.4.2 From PC to EVE via Ymodem**

Connect the PC to the EVE console with a serial cable. Run ProComm on the PC as above. After getting the EVE prompt, run xmodem:

```
Eve> entersys  
System EVE# xmodem -yrt
```

In the PC ProComm terminal window, use the upload button to transfer the sXX\_NNNN.dat to the EVE ramdisk. Then copy it to config.dat:

```
System EVE# copy sXX_NNNN.dat config.dat
```

Again, make sure you have used the correct station number. Restart EVE and watch for errors.

```
System EVE# start
```

## **5.0 Base System Programs and Scripts**

### **5.1 Packet Decode (pdecode)**

Packets produced by a PAM station must first be decoded before the data can be used by other software. The **pdecode** program provides some flexibility in decoding PAM packets.

The following manual page can be displayed on the UNIX system by entering "**man pdecode**".

**NAME**

pdecode – Decode PAM packet files

## SYNOPSIS

`pdecode`

`[-h] [-H] [-l] [-m missing_val] [-n] [-N nc_server_host] [-o output[,output[,...]]] [-p precision] [-P] [-s station_id[,station_id[,...]]] [-S] [-t time_format] [-w field_width] [-r RPC_timeout] [-b RPC_batchperiod] [input_packet_file]`

## DESCRIPTION

**pdecode** interpretes PAM packet data, and sends the decoded data variables either to NetCDF files via the `nc_server` process, or to the standard output in ASCII form.

**pdecode** can interpret packets in either PPF or hex format.

## OPTIONS

**-h**

Print a short help message concerning the runstring options.

**-H** Print a header at the beginning of ASCII output.

**-l** Print time using the local time zone. Otherwise time is printed as GMT. The local time zone can be set using the `TZ` environment variable.

**-m** *missing\_val*

Numeric value to print when data is missing. Default is 1.e37.

**-n** Send PAM data variables to NetCDF files via the `nc_server` process on the current host. If neither **-n** nor **-N** options are specified, the data will be printed on the standard output in ASCII form.

**-N** *nc\_server\_host*

Send PAM data variables to NetCDF files via the `nc_server` process on the *nc\_server\_host*. Use the standard network name for the host.

**-o** *output[,output[,...]]*

Select one or more OUTPUT variable groups to decode. The OUTPUT names and the variables in an OUTPUT group are set in the appropriate `fmtNNNN.dat` file in the `$PAM/config` directory. If the **-o** option is not specified, all OUTPUTs will be decoded and output.

**-p** *precision*

Number of digits of precision in numeric ASCII output. Default is 4.

**-P** Pass through. If NetCDF output is also selected, the encoded packets are passed on to the standard output.

**-s** *station\_id[,station\_id[,...]]*

Select one or more stations for output by `STATION_ID`, as shown in `$PAM/config/projects/$PROJECT`. If the **-s** option is not specified, packets from all stations will be decoded and output.

**-S** System log. Send error and information messages to the `syslog` process. Do "man `syslog`" for more information. If this option is not in effect, messages are sent to the standard error.

**-t** *time\_format*

Format to use when printing time on the standard output. The format is constructed of "%X" fields, in a manner similar to C `printf` formats. Do "man `strptime`" for information about time formats. The default for SunOS 4.x is "%Y %m %d %H%M%S j%j". The default for SunOS 5.x is the value of the `$CFTIME` environment variable, or if that is not defined the same format as the output of the `date` command. Some useful format fields are:

%Y 4 digit year

%y 2 digit year

%m 2 digit month (01-12)

%b month abbreviation (Jan, Feb etc)

%d 2 digit day of month (01-31)

%H 2 digit hour (00-23)

%M 2 digit minute (00-59)

%S 2 digit second (00-59)

%j 3 digit day of year (001-366)

%Z time zone abbreviation

**-w** *field\_width*

Width of ASCII numeric output. The default is 6 characters.

**-t** *RPC\_timeout*

Number of seconds to wait for response from nc\_server process. Default is 120 seconds. Should not need to be changed unless RPC timeout problems are occurring.

**-b** *RPC\_batchperiod*

Time period in seconds for checking for existence of nc\_server. Default is 300 seconds. Should not need to be changed unless RPC timeout problems are occurring.

**input\_packet\_file**

Name of packet file to read. If no packet file is specified, **pdecode** reads from the standard input.

## ASCII OUTPUT

ASCII output is printed in the following form:

[*stnI* *fmtN*] [*outputName*] *date\_time* *var1* *var2* ...

*I* is the station ID. *N* is the format number.

If one station is selected, using the **-s** option, then the *stnI* and *fmtN* fields are not printed. The format of *date\_time* is controlled by the **-t** option. *var1* are the numeric values of the variables in the OUTPUT group.

If one output is selected, using the **-o** option, then the *outputName* field is not printed. Otherwise records from the available output groups are mixed together as they occur.

## HEADER

If the **-H** option is specified, then a header for each OUTPUT is printed, in the following form:

*outputName* *name1* *name2* ...

*name1* *name2* ... are the EVE variable names from the format file.

## NETCDF OUTPUT

Remote procedure calls (RPC) are used to transfer data to the nc\_server process on the specified host. If the nc\_server process is not running on that host, an error message

hostname: RPC: Program not registered

will occur.

## EXAMPLES

Decode packets into ASCII from a file METHEX.162:

```
pdecode METHEX.162
```

Same command, using file redirection:

```
pdecode < METHEX.162
```

Decode packets from the ppoll process into NetCDF files. Errors are logged to the system log:

```
ppoll -S | pdecode -S -n
```

Output MET and RAD data as ASCII with a header showing the variable names in each. Packets are read from a file packet.dat.

```
pdecode -H -o MET,RAD packet.dat
```

Output RAD data from station 1 as ASCII. Time will be formatted as "year yday hour min sec" in the local time zone. In this example, all fields will be numeric, which may make it easier for reading by other software.

```
pdecode -s 1 -o RAD -t "%Y %j %H %M %S" packet.dat
```

## ENVIRONMENT

### PAM

The root of the PAM software tree.

### PROJECT

The current project name, e.g. GAME97 or SHEBA.

### EVE\_CONFIG

The directory path of the EVE format files. Optional. If this environment variable does not exist, the format files are assumed to exist in \$PAM/config. The station file should be known as \$EVE\_CONFIG/projects/\$PROJECT or \$PAM/config/projects/\$PROJECT.

### PAM\_NETCDF

### SURF\_NETCDF\_DIR

The path for the output NetCDF files. If PAM\_NETCDF does not exist, then SURF\_NETCDF\_DIR is checked. If neither exist the program does an exit(1).

**SURF\_NETCDF\_FILE**

The file name format for the output NetCDF files. It is usually something like "project.%y%m%d.nc". The %x descriptors are replaced by cftime when creating the NetCDF file names. If SURF\_NETCDF\_FILE does not exist, the name format is created from the project name, after converting to lower case, and removing trailing digits.

**TZ** Local timezone.

**CFTIME**

(Solaris 2.x) Format to use when printing time.

**FILES****\$PAM/config/fmtXXXX.dat**

EVE configuration file.

**\$PAM/config/\$PROJECT**

Project station configuration file.

**SEE ALSO**

**strftime(3)**, **ppoll(1)** **parch(1)** **nc\_server(1)** **strftime(3)** **cftime(3)**

**BUGS**

Report them to Gordon Maclean (maclean@ucar.edu).

## 5.2 Packet Archive (parch)

**parch** copies packets to a directory, separating them by day into files. Using **parch** ensures the packet archive is organized into directories, using a common file naming convention.

The following manual page can be displayed on the UNIX system by entering “**man parch**”.

**NAME**

**parch** – Archive PAM packets

**SYNOPSIS**

**parch**

**[-f file\_name\_format] [-S] [archive\_directory]**

**DESCRIPTION**

**parch** reads packets from the standard input, appends them onto files in an archive directory, and echoes them onto the standard output.

**OPTIONS**

**-f file\_name\_format**

Format to use when creating archive file names. The default is "%y%m%d.pam". The format is constructed of ordinary characters and "%X" fields, in a manner similar to C printf formats. Do "man strftime" for information about time formats. Some useful format fields are:

%Y 4 digit year

%y 2 digit year

%m 2 digit month (01-12)

%d 2 digit day of month (01-31)

%H 2 digit hour (00-23)

%M 2 digit minute (00-59)

%S 2 digit second (00-59)

%j 3 digit day of year (001-366)

**-S** System log. Send error and information messages to the syslog process. Do "man syslog" for more information. If this option is not in effect, messages are sent to the standard error.

### **archive\_directory**

Directory where packet archive files are placed. If it is an absolute directory path, beginning with a slash, '/', the files are placed in that directory. If it is a relative path, not beginning with a slash, it is used as a subdirectory of \$PAM/projects/\$PROJECT/packets. In either case, the directory is created if it does not exist.

### **PACKET ARCHIVE FILES**

Each packet archive file contains packets with time tags of 00:00 to 23:59 GMT on a given day. The name of the file is created by formatting the time at 00:00 GMT of the day, using the *file\_name\_format*. If the file exists, **parch** appends packets to the end of the file. Therefore, depending on the sequence that packets are processed, the packets in an archive file may not be in time sequence.

**parch** does not segregate packets by station id. All packets received from any station on a given day are stuffed into the same file.

### **EXAMPLE**

Read packets from a pipeline, and archive them in \$PAM/projects/\$PROJECT/packets/pollled. This is a typical command when polling PAM stations via an RF modem during a field project:

```
ppoll -S METPPF | parch -S polled | pdecode -S -n &
```

Read packets from a file "sta1/met169.dat" and store them in /data/packets/local:

```
parch /data/packets/local < sta1/met169.dat
```

### **ENVIRONMENT**

#### **PAM**

The root of the PAM software tree.

#### **PROJECT**

The current project name, e.g. GAME97 or SHEBA.

#### **EVE\_CONFIG**

The directory path of the EVE format files. Optional. If this environment variable does not exist, the format files are assumed to exist in \$PAM/config. The station file should be known as \$EVE\_CONFIG/projects/\$PROJECT or \$PAM/config/projects/\$PROJECT.

### **FILES**

#### **\$PAM/config/fmtXXXX.dat**

EVE configuration file.

#### **\$PAM/config/\$PROJECT**

Project station configuration file.

### **SEE ALSO**

**ppoll(1)** **pdecode(1)**

### **BUGS**

Report them to Gordon Maclean (maclean@ucar.edu).

## **5.3 Poll Pam Stations (ppoll)**

Use the **ppoll** program to fetch packets from a PAM station when a serial connection to the EVE auxiliary console is available. **ppoll** periodically sends a *pollNNN* command over the serial link, where NNN is the station id of each configured PAM station.

The following manual page can be displayed on the UNIX system by entering “**man ppoll**”.NAME

ppoll – Poll PAM stations for packet data.

### **SYNOPSIS**

ppoll

**[-a auxiliary\_program] [-b back\_poll\_days] [-B baud\_rate] [-d tty\_name] [-D data\_bits] [-h] [-L] [-n ntry] [-p polling\_period] [-P parity] [-S] [-t polling\_timeout] [-T stop\_bits] [-v] [-w warning\_period] [EVE\_FILE]**

## DESCRIPTION

**ppoll** polls PAM stations for packet data via a serial port, and writes the received packets to the standard output.

A serial port on the UNIX base system must be connected to the EVE AUXCONSOLE port on the PAM station. This can be a direct copper connection for testing in the lab, in which case only one PAM station could be polled.

Alternatively, connections to one or more remote PAM stations can be established with RF modems configured in Point-to-Multipoint mode, with the master modem connected to the UNIX base system. **ppoll** can then poll the remote PAM stations in succession.

The list of stations to be polled, the polling period, and the polling time offset for each station are contained in the project configuration file, \$PAM/config/projects/\$PROJECT.

**ppoll** uses the **pollNNN** EVE system console command, where NNN is the station id. See the EVE User's Manual for documentation on the **poll** command.

If packets have not been received from a station for a period of time **ppoll** will back-poll for packets since the last packet received, up to *back\_poll\_days* ago. On start up, **ppoll** scans the directory \$PAM/projects/\$PROJECT/packets/pollled for packet files to determine the last packet from each station. Therefore packets received from **ppoll** should be archived in the "pollled" directory, in order for the back-polling to work correctly on startup, as follows:

**ppoll** ... | **parch** polled

## OPTIONS

**-a** *auxiliary\_program*

Name and run-time arguments of an auxiliary program to be executed every polling cycle. Enclose name and blank-separated arguments in one string, delimited by single or double quotes. See AUXILIARY program section, below.

**-h** Print a short help message concerning the runstring options.

**-b** *back\_poll\_days*

If a station has not been responding, poll it for packets as much as *back\_poll\_days* days old. The default is 5 days. If you do not want to back-poll, set *back\_poll\_days* to 0.

**-B** *baud\_rate*

Baud rate of serial port connection. The default is 19200 baud.

**-d** *tty\_name*

Name of tty port. Under SunOS 4 the default is /dev/ttya. Under SunOS 5 the default is /dev/term/a.

**-D** *data\_bits*

Number of data bits. Default is 8.

**-L** Set CLOCAL bit in serial connection. If CLOCAL is set, the state of Carrier Detect (CD) on the serial port does not effect polling to the device once it is opened. Carrier Detect does effect whether the device can be opened, however. See discussion of "No such device" and "I/O error" under ERRORS section.

**-n** *ntry*

Number of times to poll a station each polling period if it does not respond. Default is 5. Must be greater than or equal to 1.

**-p** *polling\_period*

Polling period (seconds). The default is read from the POLL\_RATE column in the project configuration file, \$PAM/config/projects/\$PROJECT.

**-P** *parity*

Parity: none (default), even, odd.

**-S** System log. Send error and information messages to the syslog process. Do "man syslog" for more information. If this option is not in effect, messages are sent to the standard error.

**-t** *polling\_timeout*

Seconds to wait for a response after polling a station. Default is 5.

**-T** *stop\_bits*

Number of stop bits. Default is 1.

**-w** *warning\_period*

Hours between warnings about lack of data from a station. If the **-S** option is in effect, these warnings are written to the system log.

**-v** Verbose. Print polling conversation between **ppoll** and EVE. If the **-S** option is in effect the messages will be printed to the system log at "debug" severity.

## EVE\_FILE

Name of EVE FILE to poll. The names and contents of the EVE FILES are configured in the format file, \$PAM/config/fmtXXXX.dat.

## TIMING

Clock agreement between the PAM station and UNIX system is not critical, but **ppoll** works best when the UNIX system clock is within a few seconds of the EVE clock. If the UNIX system clock is ahead of the EVE clock, then the latest packet may not be ready when a station is polled. **ppoll** will likely then back-poll for more than one packet. This causes a bit more processing load on EVE, but should not cause problems unless the clocks are wildly out of agreement. If the UNIX system clock is behind, the packets will be behind real-time by the same amount.

## AUXILIARY PROGRAM

If the **-a** option is used, **ppoll** will close the tty port and execute another process after completing each polling cycle. An auxiliary program can be used to send EVE commands to stations based upon data in the most recent packets. This program is passed two additional arguments (after any specified with the **-a** option): the name of the tty port, and the name of a file containing the packets received from all stations during the last polling period.

The normal shell search path is used when searching for the program, which means that the directory containing the program should be in the user's **PATH** environment variable, or the name of the program should be a fully qualified path name, starting with a slash.

When it is run, the auxiliary program can open the tty port for reading and writing. The auxiliary program does not need to delete the packet file when it is done, **ppoll** will re-use it.

The auxiliary program should execute promptly and exit, so that **ppoll** can resume the polling process. Currently **ppoll** will not execute the auxiliary program if less than 30 seconds remain before the start of the next polling cycle. That means that if one or more stations are not on the air, and depending on the value of the **-n**, **-p** and **-t** options, that the auxiliary program may not be executed.

## EXAMPLES

Poll for METPPF packets, pass packets to the archive program (**parch**) and packet decoder (**pdecode**) for writing to NetCDF files. Serial port "a" is used, at 19200 baud, no parity, 8 data bits, 1 stop bit. Errors and warning messages from all programs are written to syslog:

```
ppoll -S METPPF | parch -S polled | pdecode -S -n &
```

Poll for METPPF packets. After polling each station, execute a program called **control\_sonic**, passing it the string "stn1", followed by the name of the tty port, and the name of a file containing packets received in the last polling period:

```
ppoll -a "sonic_heater stn1" METPPF
```

## ENVIRONMENT

### PAM

The root of the PAM software tree.

### PROJECT

The current project name, e.g. GAME97 or SHEBA.

### EVE\_CONFIG

The directory path of the EVE format files. Optional. If this environment variable does not exist, the format files are assumed to exist in \$PAM/config. The station file should be known as \$EVE\_CONFIG/projects/\$PROJECT or \$PAM/config/projects/\$PROJECT.

## FILES

### \$PAM/config/fmtXXXX.dat

EVE configuration file.

### \$PAM/config/\$PROJECT

Project station configuration file.

### \$PAM/projects/\$PROJECT/packets/poll

Archive directory for polled packets.

### /tmp/ppoll.TTY.pid

**ppoll** writes its process id to this file. TTY is the name of the tty device. Scripts can then shutdown polling by sending the hangup signal to **ppoll**:

```
kill -HUP `cat /tmp/ppoll.*.pid`
```

### \$PAM/scripts/port.nologin

Solaris 2.x C shell script for disabling login on a serial port, and optionally enabling software carrier detect.

## SIGNALS

### SIGHUP

Stop polling and exit.

## ERRORS

### No such device or address

The terminal device could not be opened. Check the device name. If it is correct, the problem can be due to a floating Carrier Detect (CD) on the serial connection. The best solution is to make sure that CD (pin 8) on the UNIX side is asserted (+12V) in the connection. If CD is not available on the serial device which is providing the connection to EVE, you can use a jumper box to tie either RTS (pin 4) or DTR (pin 20) to CD (pin 8) on the UNIX side. Testing under Solaris 2.5 on a Sparc 5 indicated that CD should be held either high (+12V) or low (-12V). If CD is floating in the nether regions, the above error occurred. This problem, which occurs when the device is being opened, cannot be solved by enabling software carrier detect or setting the CLOCAL bit with the **-L** option, though either of these will correct the problem of a lost carrier after the port is opened. See "I/O Error", below.

### I/O error

This occurs while polling over the serial device if the Carrier Detect pin is not asserted (+12V) and neither the **-L** option nor software carrier is in effect. Either the **-L** option or software carrier will enable polling to occur without the Carrier Detect signal. Otherwise **ppoll** will not send the EVE "poll" command if Carrier Detect is not present. If properly connected, the lack of Carrier Detect may actually signify that the link is down. Imagine that! In this case **-L** will not effect the success of the polling, it just means that **ppoll** will try anyway, and that an I/O error will not be reported. Since PAM data needs very low bandwidth, and **ppoll** will back poll for missed packets, it is not usually important if a poll did not succeed due to a temporary loss of carrier, so use the **-L** option to turn off the error messages if you're tired of them. Software carrier can be enabled on a port with the Solaris 2.x **pmadm** or **admintool** commands. A Solaris 2.x script, `$PAM/scripts/port.nologin` is provided for enabling software carrier detect on a port.

### tty lock file /var/spool/locks/LK.xxx exists. Is tip running?

**ppoll** checks to see if a lock file for that device exists on `/var/spool/locks`. If so, it aborts. These lock files are created by the UNIX commands **tip** and **kermit**, and are supposed to be deleted upon exit. If the lock file was not deleted due to the command aborting abnormally, you will have to remove the lock file by hand, perhaps as the super-user. **ppoll** does not create a lock file. It does, however, do an exclusive open on the serial device, so that **tip** and **kermit** cannot access the device while **ppoll** is running.

### Device busy

Another process is accessing the serial device, perhaps **tip**, **kermit** or **login**.

### Permission denied

The file permissions must be changed on the device. From the superuser account, do "chmod 666 /dev/term/X". You may also have to make sure login is not enabled on the port. You can use **admintool** from the super-user account to disable login on a port. A script is also provided (`$PAM/scripts/port.nologin`) for Solaris 2.x to disable logins on a port.

## SEE ALSO

**parch pdecode termio tip**

## BUGS

Report them to Gordon Maclean ([maclean@ucar.edu](mailto:maclean@ucar.edu)).

## pollup

Starts up polling. During SHEBA, this is run from the pam user's crontab, and so shouldn't have to be run directly by the user.

## 5.4 polldn

Shut down polling. Another script, `eve_console`, uses `polldn` to stop polling to allow the user to connect to an EVE console.

## 5.5 eve\_console

This script shuts down polling, then runs `tip`, to allow the user to communicate with an EVE console over the RF serial link. When the user exits `tip`, polling is restarted.

## 5.6 flash

This script reads packet files from a hard disk or FlashDisk on a PC running Windows95, which must be accessible over the network. The packets are archived in the \$PAM/projects/\$PROJECT/packets/flash directory and the variables are written to NetCDF files.

The Samba program “smbclient” is used to transfer the files from the PC. Do “man smbclient” for information on smbclient. The SHEBA PAM System Administration Manual contains information about how to configure the Unix and Win95 hosts.

```
Usage: flash [-r|-k] [-s server] [-d diskname]
          [-U user] [-p password] [-f fileprefix] [-n ndays | files...]
```

-r: remove packet files from PC if they have been successfully read

-k: keep packet files on PC

If neither -r or -k are specified, you are prompted for an answer

-s server: the network host name of the PC. The default is rocky

-d diskname: the name of the disk on the PC, usually c or d.

Upper or lower case is OK. This is the ShareName

of the disk, as configured in the Sharing... window of Win95

The default is the D disk

-U user: the name of the user. I don't think this is necessary,  
but threw it in anyway

-p password: this is the ReadOnly or Full password as set in the  
Sharing... window of Win95. If security is not a problem, you  
can set it to nothing (return) in Win95 and then you don't need  
to specify it here.

-f fileprefix: the file name prefix of the packet files on the PC

The default is metppf. File names of format metppf.JJJ will be  
transferred, where JJJ is the julian day

-n ndays: read the last n days of packet files from the PC.

-n 0 reads today's file

files: list of files to transfer separated by spaces. If you don't  
specify the -n nday option or give some file names this script  
does nothing.

## 5.7 logbook

Runs the tcl/tk logbook program, providing an online field log.

## 5.8 ncprint

Print NetCDF files in ASCII form. Here's an example which prints two variables (i.batt and i.load) from station 2:

```
cd $PAM/projects/SHEBA/netcdf
ncprint -s 2sheba.971014.nc i.batt i.load
```

Do:

```
ncprint
```

for usage information.

## 5.9 utime

A handy program for manipulating time in shell scripts. Do:

```
utime
```

for usage information. Here's an example from a C shell script which uses utime to read the previous weeks packet files:

```
@ t2 = `utime now`  
@ t1 = $t2 - (6 * 86400)      # 86400 seconds in a day  
@ t = $t1  
while ($t <= $t2)  
    set filename = `utime $t +sheba.%y%m%d.nc`  
    pdecode -n $filename  
    @ t += 86400  
end
```

## 5.10 Sun\_angle

The sun\_angle program is provided to allow an operator to printout the azimuth / elevation of the sun for a particular location (latitude / longitude) and date / time of year. This data is used to align a PAM sonic / anemometer / tower to a high degree of precision using a Theodolite surveying instrument. Normally, a PAM station is aligned using a hand-held compass which generally can provide an accuracy to within 2 degrees of true azimuth. That technique requires knowledge of the site's magnetic declination and freedom from magnetic anomalies, plus a skilled operator. With a Theodolite, alignment accuracy to within .1 degree or better is easily attained. The user is prompted by the program for the parameters it needs.

## 5.11 Satellite\_angle

The satellite\_angle program is provided to allow an operator to determine the azimuth / elevation look angle from a known earth location (latitude / longitude) to a geostationary satellite positioned at the specified longitude. It is used to prior to setting up a remote station as a means of determining how to point the GOES/GMS antenna. The user is prompted by the program for the parameters it needs.

## 6.0 Assorted UNIX Information

### 6.1 Editing Files

Several editors are available on the system, including vi, emacs and dtpad.

#### 6.1.1 dtpad editor

dtpad (desktop pad) is the easiest to use for people not familiar with UNIX. It is a typical point and click editor, with menus that provide for opening and saving files. dtpad will only run when the X window system is up, so it may not be available when doing certain system administration tasks, in which case one must use vi.

dtpad is also a bit more difficult to run when being used from a different account than the one that started X windows. For example, if you want to run the editor from the superuser account to fiddle with a system file, while the X window desktop is being run from the pam account, you must do:

```
xhost +
su -
<provide super user password>
# now running as super user
tssh
setenv DISPLAY :0
dtpad /etc/hosts
exit
```

#### 6.1.2 The joys of vi

In certain circumstances you may not be able to avoid the edifying experience of using a truly classic editor. Here's a cheat sheet that should allow you to get by.

```
***** vi cheat sheat *****
# to edit an existing or new file:
vi filename

vi Commands
h          move cursor left (or the cursor keys may work)
j or <return> move cursor down
k          move cursor up
l          move cursor right
99G       go to line 99

o          open line below cursor. Enters insert mode.
           Type characters, then ESC key to get back to command mode.
O          open line above cursor. Enters insert mode. Type ESC when done
i          insert characters in front of cursor. Type ESC key when done.
cw        change word. Current word is replaced by input. Type ESC when done.

x          delete current character
```

dd	delete current line
dw	delete word under cursor
u	undo last edit
/abc<return>	look for string "abc"
n	look for next occurrence of string
:f	show current file and line number
ZZ	save file and exit
:w!	save a readonly file (such as /etc/inet/hosts)
:q!	exit without saving file

### 6.1.3 Help I'm in ed!

By mistake you may have the misfortune of starting “ed”, a stone-age, line editor. To get out of “ed”, type “q” or ctrl-D.

### 6.1.4 emacs

Since the author is not an emacs user, you'll have to get help elsewhere...