3 Basic *AIPS* Utilities

This chapter reviews some basic *AIPS* utilities with which you should be familiar before you start calibrating data or processing images in *AIPS*. Many of these utilities will appear in later chapters on calibration, image making, and so on. However, in those chapters, these utilities will be explained only briefly.

3.1 Talking to *AIPS*

3.1.1 *POPS* and *AIPS* utilities

When using the *AIPS* system, you talk to your computer through a command processor called *POPS* (for People Oriented Parsing System) that lives in the program `AIPS`. The steps needed to start this basic program are discussed in § 2.2.3. The copy of this program that you get will be called `AIPS`*n* where *n* is often referred to as the "*POPS* number" of your session.

The *POPS* command processor is not unique to *AIPS*. It has been present in other programs at the NRAO for many years, and will be familiar to users of the NRAO single-dish telescopes. Chapters 4 to 11 of this *CookBook* give explicit examples of most of the *POPS* commands that a new *AIPS* user needs to know, so we will not give a separate *POPS* tutorial here. The command HELP POPSYM C_R will list the major *POPS* language features on your terminal, and Chapter 12 below reviews some advanced features of *POPS*.

As well as providing a command processor, *AIPS* replaces many features of your computer's operating system with its own utilities. This may seem inconvenient at first — you will have to learn the *AIPS* utilities as you go along. You will see the advantage of this approach when you use *AIPS* in a computer that has a different operating system. Your interface to *AIPS* will be almost identical on a VAX, or a Convex C-1, or a Unix-based workstation, or a Cray X-MP. Once learned, your *AIPS* skills will therefore be highly portable.

Lists of the important *AIPS* utilities can be obtained at your terminal by typing ABOUT CATALOG C_R and ABOUT GENERAL C_R. See also Chapter 13 for a relatively recent version of all such category lists.

3.1.2 Tasks

AIPS provides a way for you to set up the parameters for, and then execute, many applications programs sequentially or in parallel. The more computationally intensive programs may take many minutes, hours (or even days) of CPU time to run to completion. They are therefore embodied in *AIPS* "tasks" — programs that are spawned by the `AIPS` program to execute independently and asynchronously (unless you choose to synchronize them). This lets you get on with other work in `AIPS`, while one or more tasks are running. You may spawn, however, only one copy at a time of each task from a given *AIPS* session (*i.e.*, *POPS* number.)

A typical task setup will look like:

> TASK '*task_name*' C_R	to make *task_name* the default for later commands; note the quote ' marks.
> HELP C_R	to write helpful text on your terminal about the purpose of the task and about its input parameters.

You will then spend some time setting up parameter values, as in § 3.1.4 below. Then, type

> INP C_R	to review the parameter values that you have set and
> GO C_R	to send the task into execution.

You may also specify which task you want to execute by an immediate argument, *e.g.*, GO UVSRT C_R to execute the task UVSRT. After the GO step, you will watch for messages saying that the task has started executing normally, has found your data, etc., while you get on with other work in AIPS.

If you discover that you have started a task erroneously, you may stop it abruptly with

> ABORT C_R	to kill the task named by TASK, or
> ABORT *task_name* C_R	to kill *task_name*.

This will stop the job quickly and delete any standard scratch files produced by it. However, input data files — and output data files that are probably useless — may be left in a "busy" state in your data catalog. The catalog file is described in § 3.3, including methods to clear the "busy" states and to delete unwanted files. Note that you may also stop a task running under another AIPS number by setting adverb NUMTELL to that number.

Similarly, you may tell the current AIPS session to wait for the completion of a task before doing something else using

> WAIT C_R	to wait on the task named by TASK, or
> WAIT *task_name* C_R	to wait on *task_name*.

You may also wait on a task running under another AIPS number by setting adverb NUMTELL to that number. Note that AIPS will continue after the task finishes even if it finishes badly; see § 12.2.2.

The current full list of tasks may be obtained on your terminal (or workstation window) by typing ABOUT TASKS C_R. Since this list runs for many pages, you may wish to direct the output to the line printer (with DOCRT = -2 C_R) or to consult the list in Chapter 13 of this *CookBook*.

3.1.3 Verbs

Some of the smaller AIPS utilities run quickly enough to be run inside the AIPS program rather than being spawned. These "verbs" include simple arithmetic and POPS operations, the HELP, ABOUT, INP, and GO commands mentioned already, interactive manipulations of the TV-like display, and many more. Verbs are sent into action simply by setting their input parameters and typing the name of the verb followed by C_R. (The sequence GO *verb_name* C_R will also work, but a bit more slowly since it also saves the input parameters of the verb for you; see § 3.5 for a further discussion of saved parameters. The sequence TASK '*verb_name*' ; GO C_R will not work, however.) While a verb is executing, AIPS will not respond to anything you type on the terminal (but it will remember what you type for later use). Just watch out for messages and do what is called for with the TV cursor or terminal. You may, of course, think about what you will do next.

You can list all the verbs in AIPS on your terminal by typing HELP VERBS C_R, but the output lists only the names. To find out more, type ABOUT VERBS C_R which describes what the verbs do. Since this output fills several pages, you may wish to direct it to the line printer (set parameter DOCRT to -2), or to consult the (perhaps dated) list printed in Chapter 13 of this *CookBook*.

3.1.4 Adverbs

AIPS uses "adverbs" (which may be real numbers or character strings, scalars or arrays) to pass parameters to both "verbs" and "tasks." A significant part of your personal time during an *AIPS* session will be spent setting adverbs to appropriate values, then executing the appropriate verbs or tasks. Examples of adverb-setting commands in *AIPS* are:

> CELL 0.5 C_R	to set a single scalar `CELL` to `0.5`
> CELL 1/2 C_R	alternate for above with *POPS* in-line arithmetic
> IMSIZE 512,256 C_R	to set a two-element array `IMSIZE` to `IMSIZE(1)=512, IMSIZE(2)=256`
> IMSIZE 512 256 C_R	an alternate for the above if both values are positive
> IMSIZE 256+256,256 C_R	an alternate for the above using *POPS* in-line arithmetic
> UVWTFN 'NA' C_R	to set a string variable `UVWTFN` to the value `NA`
> LEVS 0 C_R	to set all elements of the 30-element array `LEVS` to zero
> LEVS = -2,-1,1,2,3,4,5 C_R	to set `LEVS(1)=-2, LEVS(2)=-1, LEVS(3)=1`, etc. The `=` avoids in-line arithmetic that would otherwise subtract 2 from `LEVS(1)`
> LEVS = -2,-1 1 2 3 4 5 C_R	an alternate for the above; the comma avoids in-line arithmetic that would otherwise set `LEVS(1) = -3`

Many *AIPS* tasks will assume sensible "default" values for adverbs that you choose not to (or forget to) specify. Some adverbs cannot be sensibly defaulted; these should be clearly indicated in the appropriate help information. You may review the current input parameters for any *AIPS* task or verb by typing

> INP C_R	to review the parameters for task `TASK`, or
> INP *task_name* C_R	to review the adverbs for task *task_name*.

Any adverbs which you have set to *à priori* unusable values will be followed on the next line by a row of asterisks and an informative message. Details of the input parameters used by any *AIPS* verb or task can be obtained on your terminal by typing:

> HELP C_R	to review the parameters for task `TASK`, or
> HELP *task_name* C_R	for task *task_name*
> HELP *verb_name* C_R	for verb *verb_name*
> HELP *adverb_name* C_R	for adverb *adverb_name*

See § 3.8 below for more methods of obtaining on-line help with *AIPS*.

You can list all the adverbs in *AIPS* on your terminal by typing HELP ADVERBS C_R, but the output lists only the names. To find out more, type ABOUT ADVERBS C_R which describes what the adverbs do. Since this output fills several pages, you may wish to direct it to the line printer (set parameter `DOCRT` to -2), or to consult the (perhaps dated) list printed in Chapter 13 of this *CookBook*.

3.2 Your *AIPS* message file

AIPS and all tasks talk back to you by writing messages to a disk file called the "message file" and/or by sending them to you on the appropriate "message monitor." Simple instructions and progress messages usually go only to the monitor; very few (if any) messages go only to the file. For AIPS itself, the message monitor is always the workstation window or terminal into which you are typing your commands. For the tasks, the monitor can also be a separate terminal (on well-equipped, but old, systems) or a second workstation window under control of the *AIPS* dæmon process MSGSRV. You can control whether or not you get the message server window by the setting of a Unix environment variable (AIPS_MSG_EMULATOR). Enter

> HELP MSGSRV C_R	for details.

At most *AIPS* installations, you get a message server by default. You may also control the size and appearance of the message server with parameters in the X-Windows .Xdefaults file. These parameters are also listed in by HELP MSGSRV C_R.

You may review the contents of the message file by typing PRTMSG C_R at the > prompt at your terminal. PRTMSG is an example of an *AIPS* "verb" — it does not need a GO from you to execute, and it is *not* shed from your terminal. Each message in the file has, associated with the text, the time, task name, *POPS* number, and the priority of the message. The priority codes range from 0 for user input to 2 for "unimportant" messages to 5 for "answers" and other significant normal messages to 8 for serious error messages. The PRTMSG verb has adverbs to let you select either the printer or your window or terminal for the display and to let you control which messages will be displayed. For example, to set the minimum priority level for messages to be displayed, type:

> PRIORITY *np* C_R	where *np* is the desired minimum level,

before running PRTMSG; then only messages at this level or above will be listed on the printer or terminal. If *np* is ≤ 5, then messages at level 0 are also shown. PRTMSG has further adverbs to limit the output by program name (PRTASK, uses minimum match), message age (PRTIME as upper limit to the age), and *AIPS* number (PRNUM). Note that PRNUM must be your *AIPS*, *i.e.*, *POPS*, session number, not your user identification number. The choice of the output device is made with

> DOCRT -1 C_R	to select the line printer or a text file under control of OUTPRINT.
> DOCRT 1 C_R	to select the terminal at its current width ≥ 72 characters
> DOCRT *nc* C_R	to select the terminal at width *nc* characters: $72 \leq nc \leq 132$.

The wider you can make your window display, up to 132 characters, the more information *AIPS* can put on a line. You may change the line printer selection with PRINTER. If the line printer is selected (DOCRT=-1; OUTPRINT=''), PRTMSG will ask for permission to proceed if more than 400 lines will be printed.

PRTMSG does not delete messages from your message file. Use:

> CLRMSG C_R	to delete messages and to compress the message file.

CLRMSG supports adverbs like those of PRTMSG, except that the deletion is of messages older than PRTIME and the printing is of messages younger than PRTIME seconds ago. Old messages are automatically deleted from your message file when you EXIT from AIPS. (The time limit for "old" messages is set by your local *AIPS* Manager. Usually, it is about 3 days.)

3.3 Your AIPS data catalog files

Your *uv* data sets and images are your largest inputs to, and outputs from, AIPS. A summary record of all your disk data sets (*uv* data, images, beams and temporary "scratch" data created by active tasks) is kept in your disk catalog files (one per disk). To interrogate this catalog file, use:

> INDI 0 ; MCAT C_R	to list all images on all disks, or
> INDI 0 ; UCAT C_R	to list all *uv* data sets on all disks, or
> SCAT C_R	to list all scratch files on all disks.

MCAT and UCAT list files based on the INDISK adverb, SCAT always lists all disks. M2CAT and U2CAT list files based on the IN2DISK adverb. M3CAT, U3CAT, M4CAT, U4CAT, OCAT, and UOCAT are similar for the IN3DISK, IN4DISK, and OUTDISK adverbs, respectively.

A complete listing of the catalog file, which may be printed with PRTMSG, can be generated by:

> CLRNAME C_R	to reset INNAME, INCLASS, INSEQ, INTYPE, and INDISK,
> CATALOG C_R	to generate the listing.

which will list all of your disk data sets. To limit the listing to a particular name, class, sequence number, type, and/or disk, use a combination of the adverbs INNAME, INCLASS, INSEQ, INTYPE, and INDISK. The INNAME and INCLASS adverbs allow a rather powerful wild-card grammar; type HELP INNAME C_R for details. Unless you want a hard copy, it is faster to use MCAT and UCAT, although they respond only to the INDISK adverb. A typical listing looks like:

```
CATALOG ON DISK  1
CAT USID MAPNAME      CLASS   SEQ  PT     LAST ACCESS      STAT
 18   76 3C166L50K   .IIM001.    1 MA 27-OCT-1996 22:30:18
 19   76 3C166L50K   .IBM001.    1 MA 27-OCT-1996 23:02:14
 22   76 3C166L50K   .IIM001.    2 MA 28-OCT-1996 15:30:45
CATALOG ON DISK  2
CAT USID MAPNAME      CLASS   SEQ  PT     LAST ACCESS      STAT
 22   76 1200+519    .IIM001.    1 MA 01-NOV-1996 23:50:10
 23   76 1200+519    .IBM001.    1 MA 01-NOV-1996 23:59:58
 24   76 1200+519    .QIM001.    1 MA 28-OCT-1996 00:10:10
 25   76 1200+519    .UIM001.    1 MA 28-OCT-1996 00:19:19
 28   76 1200+519    .ICL001.    1 MA 02-NOV-1996 00:35:20 WRIT
 31   76 SCRATCH FILE.IMAGR1.    1 SC 02-NOV-1996 00:35:37 WRIT
 32   76 SCRATCH FILE.IMAGR1.    2 SC 02-NOV-1996 00:35:39 WRIT
CATALOG ON DISK  3
CAT USID MAPNAME      CLASS   SEQ  PT     LAST ACCESS      STAT
  2   76 3C138 A C   .UVSRT .    1 UV 22-OCT-1996 12:56:50
 36   76 1200+519    .UVXY  .    1 UV 02-NOV-1996 00:32:50 READ
 37   76 1200+519    .IMAGR .    1 UV 02-NOV-1996 00:34:25 WRIT
```

This user (identification number 76) has eight image files, three on disk 1 and six on disk 2. He also has two sorted *uv* data sets and an IMAGR *uv* work file on disk 3. There are two scratch (temporary) files on disk 2 which were created by IMAGR running out of AIPS1 (this determines their IMAGR1 classname). Image data files (images and beams) are distinguished by the type code MA. The *uv* data files are distinguished by the type code UV and scratch files by type SC.

Note that this user has encoded useful information other than the source name into the image file names on disk 1. These images were of 3C166 at L band with 50 kilo-wavelength (*uv*) taper. Such information is also carried in AIPS history files (see § 3.4 below), but it is often useful to place it at a level where CAT can see it. The user also gave the UVSRT file in slot 2 on disk 3 a name that encodes the source name (3C138), the VLA configuration (A), and the observing band (C). Careful choice of AIPS file names can save much other bookkeeping. The file name can be any valid string up to 12 characters long. Also note how SEQ numbers distinguish different versions of a file with the same name; this and the global variables in AIPS are helpful features when doing iterative computations such as self-calibration.

3.3.1 Speedy data file selection

Each catalog entry has an identification number called the "catalog slot number". The CAT column at the left of the listing above shows these catalog numbers. They can be used to set up inputs quickly for AIPS programs that read cataloged disk data sets. Use:

> INDI *n1* ; GETN *ctn1* C_R	where *n1* selects the disk and *ctn1* is the catalog slot number.

The verb GETNAME (abbreviated through minimum match as GETN above) sets the adverbs INNAME, INCLASS, INSEQ, and INTYPE used by many tasks and verbs. Some tasks require a second, a third and even a fourth set of input image name adverbs. For these, use:

> IN2D *n2* ; GET2N *ctn2* C_R	to set the second set, and
> IN3D *n3* ; GET3N *ctn3* C_R	to set the third set.
> IN4D *n4* ; GET4N *ctn4* C_R	to set the fourth set.

The verb GETONAME (GETO for minimum match) sets the adverbs OUTNAME, OUTCLASS and OUTSEQ to those of a pre-existing output file. GETO is particularly useful with calibration tasks that copy extension tables (*e.g.*, CL or FG tables) from one database to another or for restarting an image deconvolution.

3.3.2 Catalog entry status

Note that several catalog slots on disks 2 and 3 in our sample catalog listing above do not have blank entries in the STAT column. This listing was made while the user was running a Clean deconvolution with IMAGR on the sorted *uv* data set in slot 36 — this *uv* data file is opened for READing. The Clean image file, ICL001 in slot 28, and the scratch and IMAGR files are opened for WRITing. Procedures that attempt to read files which are opened for writing, or vice versa, will be rejected with appropriate error messages. You must therefore note any non-blank entries in the STAT column carefully. In some situations (mainly involving system crashes or abortion of tasks [§ 3.1.2]) files may be left in READ or WRIT status indefinitely. If this happens, you may reset the file status with CLRSTAT C_R after issuing the appropriate INDISK and GETNAME. Note that a WRIT status on a file which is not, in fact, being used at present probably indicates that the data in the file have been corrupted. Such files should usually be removed from your catalog by first clearing the file status with GETN *nn*; CLRST C_R then deleting them with ZAP C_R.

Before using a data set as input to an AIPS task, check that the data set has a clear status. (It is possible to let two tasks read the same data at the same time, but this is not recommended as it will usually slow execution.) Also note the data set's disk number and its ordinal number in the catalog, as these are useful for GETN, GET2N, etc.

3.3.3 Renaming data files

Files may be renamed, after they have been cataloged, using the AIPS verb RENAME. Typical inputs might be:

> INDI 2 ; INNA '1200+519' C_R	to select disk 2 and set the input (old) name.
> INCL 'IIM001' ; INSEQ 1 C_R	to set the rest of the input name adverbs, *i.e.*, to select the file in slot 22 on disk 2 in the example above.
> OUTN '1200+51 15K' ; OUTSEQ 2 C_R	to set desired output name and sequence number.

> INP RENAME C_R	to review the inputs.
> RENAME C_R	to rename the I image to '1200+51 15K' and reset its sequence number to 2.

Two verbs can be used to alter the catalog numbers of files. RENUMBER moves a file to an empty, user-specified slot; a one-line command to do this would be SLOT *n*; RENUM C_R where *n* is the new slot number. Note that *n* may now be higher than any slot numbers currently in use. RECAT compresses the catalog (*i.e.*, it removes gaps in the catalog numbers) without changing the order of the entries in the catalog.

3.3.4 Header listings

Every image or *uv* data set in AIPS has an associated header file that contains information needed to describe the data set in detail.

The header also contains information on the number of extension files of each type that have been associated with the data set. The most important file extensions that can be associated with AIPS image data are the HIstory file described below, the CC or Clean component files (see Chapter 5) and the PLot files and SLice files (see Chapter 6).

Multi-source *uv* data files may have many extensions (see Chapter 4). The most important are the HIstory file, the ANtennas file (subarray geometric data, date, frequency and polarization information, *etc.*), the BP (bandpass) file for bandpass calibration data, the CL (calibration) file for calibration and model information, the FQ (frequency) file for frequency offsets of the different IFs, the FG (flag) file for editing information, the NX (index) file (which assists rapid access to the data), the SN (solution) file for gain solutions from AIPS calibration routines, and the SU (source) file with source-specific information such as name, position, and velocity. Chapter 4 describes the use of these extensions in some detail.

You can list the header file of any catalog entry on your terminal by following the GETNAME step above with

> IMHEAD C_R	for a detailed listing, or
> QHEADER C_R	for a shorter listing.

The output of IMHEAD and QHEADER can also be printed using PRTMSG (at PRIORITY 2).

Output from IMHEAD on a multi-source *uv* data set might look like:

```
Image=3C345     (UV)         Filename=Z17G1_A     .MULTI .   1
Telescope=SBLNKGYO           Receiver=VLBI
Observer=FAP                 User #= 1353
Observ. date=27-FEB-1991     Map date=13-JUN-1995
# visibilities    112813     Sort order  TB
Rand axes: UU-L  VV-L  WW-L  BASELINE  TIME1  WEIGHT  SCALE
           SOURCE
--------------------------------------------------------------
Type    Pixels   Coord value  at Pixel    Coord incr   Rotat
COMPLEX      1   1.0000000E+00    1.00 1.0000000E+00    0.00
STOKES       4  -1.0000000E+00    1.00-1.0000000E+00    0.00
FREQ       128   2.2228990E+10   63.50 5.0000000E+05    0.00
RA           1    16 41 17.608    1.00      3600.000    0.00
DEC          1    39 54 10.820    1.00      3600.000    0.00
--------------------------------------------------------------
Maximum version number of extension files of type SU is   1
Maximum version number of extension files of type CL is   3
```

```
Maximum version number of extension files of type HI is   1
Maximum version number of extension files of type AN is   1
Maximum version number of extension files of type NX is   1
Maximum version number of extension files of type FG is   1
Maximum version number of extension files of type SN is   1
```

Output from IMHEAD on an image file might look like:

```
Image=3C219     (MA)        Filename=3C219-BC-6  .ICL001.   1
Telescope=VLA               Receiver=
Observer=BRID               User #=   76
Observ. date=06-SEP-1992    Map date=18-APR-1994
Minimum=-1.89720898E-04     Maximum= 5.05501366E-02 JY/BEAM
---------------------------------------------------------------
Type    Pixels   Coord value  at Pixel    Coord incr   Rotat
RA---SIN   510    09 17 50.662  263.00     -0.300000    0.00
DEC--SIN   640    45 51 43.555  294.00      0.300000    0.00
FREQ         1   4.8726000E+09    1.00 2.5000000E+07    0.00
STOKES       1   1.0000000E+00    1.00 1.0000000E+00    0.00
---------------------------------------------------------------
Map type=NORMAL             Number of iterations=   50000
Conv size=   1.40 X   1.40   Position angle=   0.00
Observed RA   09 17 50.600    DEC  45 51 44.00
Maximum version number of extension files of type HI is   1
Maximum version number of extension files of type PL is   5
Maximum version number of extension files of type SL is   1
```

Both QHEADER and IMHEAD list the maximum version numbers of the table extension files associated with a data set. Because you may acquire many versions of such tables during calibration, these verbs are often invoked during calibration in AIPS. Verbs IM2HEAD. IM3HEAD, IM4HEAD, and IMOHEAD, as well as Q2HEADER, Q3HEADER, Q4HEADER, and QOHEADER appeared to make comparable listings for the second, third, and fourth input and the output file naming adverb groups, respectively.

3.4 Your AIPS history files

Every *uv* and image file has an associated "history", or HI, file. This HI "extension" of the data set stores important information about the processing done so far on the data in the file. Every AIPS task and verb that alters either the data or the file header will record its key parameters in the history file. The history file is written to the disk file when you use FITS format, so you can preserve it for reference in later AIPS sessions or when sending data to colleagues.

In general, each "card" in the history file begins with the task or verb name. It then gives one or more of the input adverb values it used (*i.e.*, the defaults are filled in). All or parts of the file may be displayed on your terminal or printed on the line printer. For example, use:

> INDISK *n* ; GETN *ctn* C_R	to select the file to be displayed.
> PRTASK 'IMAGR' C_R	to examine only history information from IMAGR.
> DOCRT 1 C_R	to direct the display to your terminal, using its full width.
> PRTHI C_R	to print the IMAGR history.
> PRTASK ' ' ; DOCRT FALSE C_R	to select all history cards and direct the output to the line printer.
> PRTHI C_R	to print the full history file.

There are several (legitimate) reasons why you might wish to edit your history files. Repetitive self-calibration cycles, or image combinations, can lead to very long and very repetitive histories which could be substantially shortened with no real loss of information. Also some entries in the history file may become obsolete by, say, the deletion of plot files. The verb `STALIN` allows you to send a range of history lines to Siberian salt mines (*i.e.*, delete) by number with optional selection by task and, optionally, interactive confirmation of each deletion. You may, of course, simply wish to add information to the history file. The verb `HINOTE` can be used to append one line, given by the adverb `COMMENT`, or many lines, typed in interactively, to the history file. Even more powerfully, the verb `HITEXT` allows you to write your history file to an external text file (see § 3.10.1). You may edit that file with your favorite Unix file editor and then read it back, writing your edited file into any AIPS history file you want (with verb `HINOTE`).

3.5 Saving and restoring inputs

All input and output parameters ("adverbs") are global throughout `AIPS`. When an adverb value is specified for, or set by, a task or verb, it remains at that value for any other task or verb that uses an adverb of the same name (until you change it). This global nature of the AIPS adverbs is useful in most cases. It can, however, be inconvenient — especially if you are taken by surprise because you have not reviewed the adverb values before running a task. Before running any task or verb, check your current input adverbs carefully with:

> INP *name* C_R	where *name* is the program name, or
> INPUTS *name* C_R	to write the input values to the message file.
> QINP C_R	to resume the previous `INP` or `INPUTS` with the page last displayed.

Some tasks have multiple pages of input parameters. `QINP` allows you to change a parameter on a page, review that page and then go on to the next page without having to view the first pages over again. Some verbs and a few tasks have *output* adverbs. Unless they are also used on input, they will not appear when you do `INP` or `INPUTS`. After running such verbs and tasks, do

> OUTPUTS *name* C_R	to view the output values and write them to the message file.

To reset all adverbs for a particular task or verb to their initial values, without changing any other adverbs or procedures, enter

> DEFAULT *name* C_R	to reset the values for *name*.
> DEFAULT C_R	to reset the values for the verb or task named in the `TASK` adverb.

You can save all adverbs you have specified for AIPS to disk at any time by typing:

> SAVE *aaaaa* C_R	where *aaaaa* is any string of up to 12 characters.
> GET *aaaaa* C_R	will restore these inputs later.

These commands save or restore your entire `AIPS` "environment". For this reason, `GET` must be the only command on the input line; `SAVE` may appear with other commands, but will be executed before *any* of the other commands on the line. Thus, the sequence INNAME '3C123' C_R INNAME 'BLLAC' ; SAVE BLLAC C_R will save a 3C123 environment, not a BLLAC one. `AIPS` automatically saves your environment in a disk area called `LASTEXIT` whenever you use the `EXIT`, `KLEENEX`, or `RESTART` commands. The command `GET LASTEXIT` is automatically executed whenever you start up the `AIPS` program again on the same machine. Thus, you retain your own AIPS environment from one use of `AIPS` to the next. To obtain a null version of the adverb values and of the rest of the AIPS environment, type:

> RESTORE 0 C_R

There is also one temporary area for saving your `AIPS` environment. To save your inputs temporarily, type:

> STORE 1 C_R	to save your inputs in area 1, and

> RESTORE 1 C_R — to recover the inputs you previously stored in area 1.

When new verbs and adverbs are created at your site, your old SAVE files will not know about them. Beginning with the 15JAN96 release, you may update the old files with the sequence:

> GET *aaaaa* C_R — to recover the old SAVE area.

> COMPRESS C_R — to get the new basic vocabularies without losing your adverb values and procedures.

> SAVE *aaaaa* C_R — to save the updated area for later; use the full name of the SAVE area here.

The list of SAVE areas may be reviewed with the verb SGINDEX. A SAVE area may be written as a RUN file (§ 3.10.2) if you first GET the area and then use SG2RUN.

The input adverb values associated with a task or a verb can be stored by the command:

> TPUT *name* C_R — where *name* is the verb, task, or procedure name.

and retrieved by the command:

> TGET *name* C_R

TPUT and TGET allow you to avoid, to some extent, the global nature of the adverb values in AIPS. This is sometimes advantageous. Whenever a task (or a verb, for that matter) is executed by the verb GO, TPUT runs automatically. TGET will therefore recover the last set of input adverbs used to execute the task, unless you deliberately overwrite them with a TPUT of your own. Note that AIPS will complain if you try to TGET input adverbs for a task for which no TPUT has previously been run (either manually or automatically). You must "put" before you can "get." TGINDEX will show you what tasks have been TPUT and when. VPUT, VGET, and VGINDEX allow you to save, recover, and list task-specific adverbs from up to 35 completely user-controlled storage areas. All TGET and VGET files have a new format which includes adverb names. Then, when the adverbs to a task or verb change, TGET and VGET will still work. In 31DEC23, the command TUGET will do a TGET but from a different user number. This may be convenient when the user uses multiple user numbers on his/her computer.

PLGET lets you recover the adverbs used to construct a user-selected plot file. As of 31DEC23, these are known by name. Older plot files depend on the list of adverbs having not changed. Check the INPUTS carefully after a PLGET.

You can change between versions of AIPS software once you are inside AIPS by typing

> VERSION '*version*' C_R — where *version* is one of OLD, NEW or TST

Alternatively, you may use this command to access a private version of a program in some other area — see § 12.2.2. Note that toggling between different versions of AIPS is possible only when the data formats are the same. Unfortunately, recent versions are not fully compatible with previous versions of AIPS. (The antenna file format changed in 2009, the internal *UV* format changed in 2015.) Note also, that you are toggling between different versions of tasks, not the verbs within the AIPS program. That version is selected when you start the program (§ 2.2.3) and can be changed only by exiting and start anew.

3.6 Monitoring disk space

Since the 15APR92 release of AIPS, the availability of data areas via NFS has vastly increased the amount of disk space accessible from a given AIPS session. The da= command line option to the aips command allows you to specify "disks" (data areas) from many hosts in addition to the current host, subject to a maximum of 35 disks per session. Note, however, that the BADDISK adverb has a limit of 10 disks. Thus, if more than 10 disks are accessed via NFS, you will not be able to prevent one or more from being used for scratch files. This can be important. Reading data over NFS is relatively efficient, but writing data is not. Even file creations (under Unix) require the writing of zeros to the whole file in order to guarantee later access to the requested space. Over NFS, this can be a slow process. For example, if user disk 1 is accessed via NFS, then every line of the message file must be written with NFS, a process which has been observed to require about one second of real time per message!

Another aspect of the new disk allocation system is a scheme by which the local AIPS Manager may restrict the availability of some disk areas to a set of user numbers, specified on a disk-by-disk basis. Managers usually use this tool to set aside most disks on a staff member's workstation for his/her sole use and to reserve space for visitors or other special projects on "public" workstations on a case-by-case basis. Use the FREESPAC verb within AIPS to show you the space used and available on all disks for your session and also to show whether or not that space is reserved. The right-most column of FREESPAC's output will show Alluser if the space is not reserved, Resrved if you are one of the users for which the space is reserved, Not you if you are not allowed to use the space, and Scratch if the space is to be used only for scratch files. Use FREESPAC often to keep track of how much space is available and where the space can be found.

Disk space is still generally at a premium. If more than one user has access to the disk areas you are using, or you use more than one user number to separate projects, then another useful tool for monitoring disks is the AIPS task called DISKU. To run it, type

> USER 32000 ; INDISK 0 C_R	to get all disks and users.
> BADDISK 0 C_R	to allow all disks; can be used to skip chosen disk numbers in 31DEC23.
> DOALL 0 ; GO DISKU C_R	to run the AIPS disk user task.

This will (eventually) list on the AIPS monitor (and the message file) the amount of data space in use by each user for all AIPS disks. Identify the worst disk hogs and apply appropriate peer pressure. If you are, mysteriously, the culprit on some disk, then

> USER 0 ; INDISK n C_R	where n is the mysteriously eaten disk
> DOALL m ; GO DISKU C_R	to run the job, listing all catalog entries requiring more than m Megabytes.

will give you the size of your larger files on the specified disk. Armed with this information, you may be able to take appropriate action upon your own data.

Sometimes the available disk space has been eaten up by AIPS scratch files that are no longer in use. Tasks that abort while executing (and other mysterious events) may produce this situation. To delete all your scratch files, except those for tasks which are still running, type:

> SPY C_R	to see which tasks are running.
> SCRD C_R	to delete the files.

SCRDEST is run automatically whenever EXIT, KLEENEX, RESTART, or ABORT *task_name* are executed. Note that the imaging and deconvolution tasks IMAGR, APCLN, and VTESS, the data editor TVFLG, the sorter UVSRT and other tasks may create large scratch and "work" files, so you should watch for "dead" copies of scratch and work files from these programs in your disk catalog. Both MCAT and UCAT will show scratch files as well as the requested file type, but SCAT will be easier to use. Note too that, if you are using more than one computer on a given disk area, only those scratch files created by your current computer will be deleted when you run the SCRD verb. Work files have to be deleted individually since they can be still of use after the task which created them has finished.

Chapter 11 of this *CookBook* tells you how to backup or delete your own data to relieve disk crowding. At present, all other methods for managing disk space involve system-dependent commands of one sort or another. Since these may have unexpected consequences they are not recommended.

3.7 Moving and compressing files

Two *AIPS* tasks are frequently used to move files from one disk to another with options to reduce the file size. They are SUBIM, used on images, and UVCOP, used on *uv* data sets. SUBIM uses the adverbs BLC and TRC to select a portion of the input image and XINC and YINC to select a pixel increment through the portion. If these adverbs are defaulted (set to 0), the entire image is copied. Clean component, history, and other table extension files are copied as well, but plot and slice extensions are not. Similarly, UVCOP uses a wide range of adverbs to select which IFs, channels, frequency IDs, times, antennas, and sources are to be copied. If all of these adverbs are defaulted (set to 0 or blank), then all data are copied except (optionally) for completely flagged records. A flag table may also be applied to the data. including flag tables too large to be handled by most tasks. With extensive data editing, UVCOP may produce a rather smaller data set even with no other selection criteria. Antenna, gain, and other table extension files are copied, but plot files are not. The task MOVE may be used to copy all files associated with a catalog number (without modification) to another disk or to another user number.

3.8 Finding helpful information in *AIPS*

Much *AIPS* documentation can be displayed on your terminal by typing HELP *word* C_R, where *word* is the name of an *AIPS* verb, task or adverb. The information given will supplement that given in the INPUTS for a verb or task. It is the only source of information on the adverbs. Type XHELP *word* C_R to display the help file in your WWW browser with links to adverbs from task help files.

To print the HELP information on your line printer, set DOCRT = -1 and enter EXPLAIN *word* C_R instead. (Using DOCRT = 1 with EXPLAIN will send the output to your terminal screen.) For the more important verbs and tasks, EXPLAIN will print extra information, not shown by HELP about the use of the program, with detailed explanations, hints, cautions and examples.

HELP may also be used to list the names of all *POPS* symbols known to AIPS by category, an operation helpful when you can't remember the name of something. Type:

> HELP ADVERBS C_R	to get a list of all adverbs in the symbol table
> HELP ARRAYS C_R	to get a list of all array adverbs in the symbol table
> HELP REALS C_R	to get a list of all real adverbs in the symbol table
> HELP STRINGS C_R	to get a list of all character string adverbs in the symbol table
> HELP VERBS C_R	to get a list of all verbs, pseudo-verbs, and procedures in the symbol table
> HELP PSEUDOS C_R	to get a list of all pseudo verbs in the symbol table
> HELP PROCS C_R	to get a list of all procedures in the symbol table

In the past, *AIPS* contained a range of general HELP files which purported to list all verbs and tasks in various categories. Since these were maintained by hand, they were essentially never current and complete. That entire system has been replaced by the verbs ABOUT and APROPOS to be discussed below. A few general help files do remain, and they may even be relatively current. A list of these may be found by typing:

> HELP HELP C_R	for help on HELP.

A few general help files remain. They are POPSYM (symbols used in *POPS* interpretive language), NEWTASK (writing and incorporating a new task into *AIPS*), and PANIC (solutions to common problems).

The HELP verb is very useful, but only if you know that the function you want exists in *AIPS* and know its name. Two functions have appeared in AIPS to assist you in this search. The first of these, APROPOS, searches all of the one-line summaries and keywords of all *AIPS* help files for matches to one or more user-specified words. For example, type

> APROPOS CLEAN C_R	to display all keyword and 1-line summaries of help files containing words beginning with "clean" (in upper and/or lower case), and
> APROPOS 'UV PLOT' C_R	note the quote marks which are required if there are embedded blanks, or
> APROPOS UV,PLOT C_R	to display all keyword and 1-line summaries of help files containing *both* words beginning with "uv" and words beginning with "plot."

The text files used by APROPOS are maintained by the *AIPS* source code maintenance (check-out) system itself. As a result, they should always be current. Of course, the quality of the results depends on the quality of the programmer-typed one-line and keyword descriptions in the help files. These were not regarded as important previously and hence are of variable quality.

The second new method for finding things in *AIPS* is the verb ABOUT. Type

> ABOUT *keyword* C_R	to see a list of all *AIPS* tasks, verbs, adverbs, etc. which mention *keyword* as one of their "keywords."

You need only type as many letters of *keyword* as are needed for a unique match. The source-code maintenance system is used to force all help files to use only a limited list of primary and secondary keywords. Software tools to update the list files have also been written, and are used at least once with every *AIPS* release. The list of categories recognized is as follows (where only the upper-case letters shown in the name are actually used):

```
ADVERB        POPS symbol holding real or character data
ANALYSIS      Image processing, analysis, combination
AP            Tasks using the "array processor"
ASTROMETry    Accurate position and baseline measurements
BATCH         Running AIPS tasks in AIPS batch queues
CALIBRATion   Calibration of interferometer uv data
CATALOG       Dealing with the AIPS catalog file
COORDINAtes   Handling image coordinates, conversions
EDITING       Editing tables, uv and image data.
EXT-APPL      Access to extension files (tables)
FITS          FITS format for data interchange
GENERAL       General AIPS utilities
HARDCOPY      Creating listings and displays on paper
IMAGE-UTil    Utilities for handling images
IMAGE         Transforming of images
IMAGING       Creation of images: FFT, Clean, ...
INFORMATion   General lists and user help functions
INTERACTive   Functions requiring user interaction
MODELING      Model fitting to uv or image data
OBSOLETE      Functions slated for removal
ONED          Functions for one-dimensional image slices
OOP           Tasks coded with object oriented principles
OPTICAL       Functions of interest for optical astronomy data
PARAFORM      Skeleton tasks for use in building new tasks
PLOT          Displays of image and uv data
POLARIZAtion  Calibration, analysis, display of polarization
POPS          Aspects of the AIPS' user language POPS
PROCEDURe     Creation of and available procedures
PSEUDOVErb    Pseudo-verbs in the POPS language and AIPS
```

```
RUN           Creation of and available RUN files
SINGLEDIsh    Functions of interest for single-disk radio data
SPECTRAL      Functions for spectra-line and other 3D data
TABLE         AIPS table extension files
TAPE          Use of magnetic tapes (obsolete)
TASK          AIPS tasks - available asynchronous functions
TV-APPL       Tasks using the TV display
TV            Basic functions on the TV display
UTILITY       Basic functions on tables, uv and image data
UV            Functions dealing with interferometer uv data
VERB          Synchronous functions inside the AIPS program
VLA           Functions of particular interest for the VLA
VLBI          Functions of particular interest for very long
              baseline data.
```

A variety of synonyms are also recognized. Besides those that are merely spelling variants, the currently accepted synonyms are

```
FILES       ->  CATALOG           POSITION   ->  COORDINATES
FLAGGING    ->  EDITING           EXTENSION  ->  EXT-APPL
PRINTING    ->  HARDCOPY          PRINTER    ->  HARDCOPY
MAP         ->  IMAGE             MAP-UTIL   ->  IMAGE-UTIL
MAPPING     ->  IMAGING           LANGUAGE   ->  POPS
CUBE        ->  SPECTRAL          LINE       ->  SPECTRAL
VISIBILITY  ->  UV                VLBA       ->  VLBI
PARAMETERS  ->  ADVERB            HELPS      ->  INFORMATION
SLICE       ->  ONED
```

More detailed descriptions of new developments in *AIPS* can be found in the *AIPSLetter* published by the NRAO every six months and tied to each *AIPS* software release. It is available from the web site `http://www.aips.nrao.edu/` and is included with *AIPS* distributions. An *AIPS* Memo series is published by the NRAO with details of various aspects of the implementation of, and planning for, *AIPS*. Advanced users may also wish to receive, and contribute to, the *AIPS* electronic mail forum — `BANANAS`. There is also an electronic news group called `alt.sci.astro.aips` devoted to *AIPS* matters. This *AIPS CookBook*, many of the *AIPS* Memos, and various other publications of the *AIPS* group are available via anonymous `ftp` (at `ftp.aoc.nrao.edu`) and via the Internet and the "World-Wide Web" starting with "URL" (Universal Resource Location) `http://www.aips.nrao.edu/`).

Your local *AIPS* Manager probably receives the *AIPSLetter*, *AIPS* Memos, and `BANANAS` and can make information from them available at your site. He/she should also be aware of the electronic means of information retrieval, and be able to help you use them. If this is not the case, write to the *AIPS* Group (at NRAO, P. O. Box O, Socorro, NM 87801-0387) or send electronic mail to **daip@nrao.edu** for further information about these services.

3.9 Magnetic tapes

Large volumes of data were once brought into, and taken away from, *AIPS* using magnetic tape. Disk files are now much more frequently used; see § 3.10. Tape usage in *AIPS* is described here, in case you still have need of it, but the descriptions of tapes elsewhere in this *CookBook* have been greatly reduced. The tape drives assigned to you are displayed as you start up AIPS, *e.g.*,

```
Tape assignments:
Tape 1 is IBM 9-track model 9348-012 on LEMUR
Tape 2 is HP 9-track model 88780B on LEMUR
Tape 3 is IBM 7208/001 Exabyte 8200 (external) on LEMUR
Tape 4 is ZZYZX 1.3Gb DAT (left, Model# ZW/HT1420T-CC6) on LEMUR
Tape 5 is ZZYZX 1.3Gb DAT (right; both 150mb personality) on LEMUR
Tape 6 is IBM Exabyte 8200 (internal) on LEMUR
Tape 7 is REMOTE
Tape 8 is REMOTE
```

for the heavily loaded, and now obsolete, IBM called lemur. The tape numbers you see above correspond to AIPS adverb INTAPE values of 1, 2, 3, and so on. The description is meant to give you some idea of which box or slot is to receive your tape. Most of the drives will have a label on them identifying their *AIPS* tape number. If in doubt, ask a local guru for help. The last two tape "drives," called REMOTE, will be discussed separately below.

In case you forget this list, the verb TAPES will show it to you. TAPES is even capable of going out on the Internet and asking what devices are available to an *AIPS* user at the computer specified by the REMHOST adverb (if it is running TPMON)! Note however that instances of TPMON are no longer started automatically when AIPS is started.

3.9.1 Hardware tape mount

On some *AIPS* systems, tapes are handled by designated operators. Before mounting tapes, read Appendix Z (for NRAO sites) or obtain directions from your local *AIPS* Manager or operators for methods by which tapes are to be handled. Most *AIPS* systems, however, are on the self-service plan. In that case, the simplest thing to do is to find a drive of the required type without a tape in it. There is no way in most Unix systems (certainly not in Linux or Mac OS/X) of reserving a tape drive globally for your exclusive use, though once you have it MOUNTed from within AIPS, no other AIPS user can access it. It is most efficient to use a tape drive directly connected to your computer (and hence listed as you started up AIPS). However, any "AIPSable" drive will do. Mount the tape physically on the drive following the mounting instructions in Appendix Z or those posted at your installation for the particular kind of tape drive. For half-inch (nine-track) tapes, don't forget to insert a write ring if you intend to write on the tape or to remove any write ring if you intend only to read the tape. Exabyte and DAT tapes have a small slide in the edge of the tape which faces out which takes the place of the write ring of 9-track tapes. For 8mm (Exabyte) tapes push the slide to the right (color black shows) for writing and to the left (red or white shows) for reading. With 4mm DAT tapes, the slide also goes to the right for writing (but white or red shows) and to the left for reading (black shows). Note the identification number m marked on the drive you are using, as you will need to provide that number to the software for mounting and dismounting the tape and for executing *AIPS* tasks which read or write tape.

3.9.2 Software mounting local tapes

After you have the tape physically mounted on the tape drive, *AIPS* must also be told that you have done this and which tape drive you have chosen. This step is called a "software tape mount." It is necessary to wait until the mechanism in the drive has "settled down", *i.e.*, when the noises and flashing lights have stopped, before you can do the software mount. This operation is done from inside `AIPS` by typing:

> INTAPE *m* C_R	to specify the drive labeled *m*.
> DENSITY *dddd* C_R	to set the density to *dddd* bpi if needed.
> MOUNT C_R	to mount the tape in software.

Read any messages which appear on your terminal carefully since they report the success, failure, and/or limitations of the operation. The meaning of "density" with modern magnetic tape devices is mostly a matter of convention. With half-inch, 9-track tapes, *AIPS* understands the usual 800, 1600, and 6250 bytes per inch densities. A special value for density, 22500, is taken to mean high density (5-Gbyte) mode on 8mm (Exabyte) tapes. You must set the `DENSITY` adverb to one of these magic values, but in many cases it does not matter which one you use.

Please dismount the tape as soon as you are finished with it, using:

> INTAPE *n* ; DISMO C_R	to dismount a tape from the drive labeled *n*.

The dismount verb should cause the tape to be rewound and, in most cases, ejected from the drive. Please remove the tape from the tape drive promptly so that others may use the drive. Note that exiting `AIPS` under most circumstances — even with `CTRL C` — will cause your mounted tapes to be dismounted automatically.

3.9.3 Software mounting REMOTE tapes

On all *AIPS* systems, the last two tape drives are indicated as `REMOTE`. This means you can use two additional adverbs in `AIPS` to access tape drives on other computers. It doesn't matter where the computer is, as long as it's connected via Internet and has *AIPS* installed on it in the conventional way. For example, if you wanted to use *AIPS* tape drive 2 on remote host `rhesus`, you would type:

> REMHOST 'RHESUS' ; REMTAPE 2 C_R	
> DENSITY *dddd* C_R	to set the density to *dddd* bpi if needed.
> INTAPE *n* ; MOUNT C_R	set local "tape" number and software mount

where *n* is the number of one of the REMOTE tape assignments in the list of tape drives you see on `AIPS` startup. If you know which computers are to provide remote tape services for you, it is a good idea to specify them when you start `AIPS` using the `tp=`*hostname* option (see § 2.2.3). In this way, you make certain that the *AIPS* dæmon tasks `TPMON`*n* which provide the remote service are running where they are needed. Note that this option is commented out of `START_AIPS` in `31DEC24`.

3.9.4 Using tapes in *AIPS*

AIPS provides a number of basic tools for managing magnetic tapes. It is very helpful to have a list of the contents of magnetic tapes you intend to read. To list the contents of a tape on the line printer:

> TASK 'PRTTP' ; INP C_R	to review the inputs.
> NFILES 0 C_R	to list all files on the tape.
> PRTLEV 0 C_R	to list the image headers but not the details — both more and less detailed listings are available.
> DOCRT FALSE C_R	to print on the line printer.
> GO C_R	to run the task.

It is also a good idea to run PRTTP on your data tapes after you have written them, but before you have deleted the data from disk. PRTTP reads the the tape record by record to test for tape errors as well as to check the data format.

The AIPS program has a number of verbs to position and check magnetic tapes. These include

> REWIND C_R	to rewind the tape, *e.g.*, after running PRTTP.
> NFILES *n* ; AVFILE C_R	to advance the tape $n > 0$ file marks.
> NFILES -*n* ; AVFILE C_R	to move the tape backwards to the n^{th} previous file.
> NFILES 0 ; AVFILE C_R	to position the tape at the start of the current file.
> AVEOT C_R	to advance the tape to the end of information, usually for the purpose of adding more data at the end.
> TPHEAD C_R	to display the contents of the data file at the current tape position.

Users are encouraged to treat magnetic tapes with some caution. The tapes themselves can have — or develop — errors which render the data in the file unavailable. Furthermore, there are no generally accepted standards governing magnetic tape software in the industry. As a consequence, each Unix operating system handles them differently and each can change over time. This creates great difficulties in AIPS and may cause your version not to handle all tape devices in a fully compatible manner.

AIPS tasks are still able to handle tapes, should the need arise. Output tapes may be written with FITTP, FITAB, and TCOPY. Input tapes are read by IMLOD, UVLOD, FITLD, FILLM, M3TAR, MK3IN, and MK3TX.

3.10 AIPS external disk files

AIPS maintains a wide range of disk files for its own use internally. Unless you intend to write programs for AIPS you need not be concerned about their formats or, in many cases, even their existence. However, recent versions of AIPS also support "external" disk files to be read from and written to disk directories controlled by you. You may read and write from/to binary "FITS-disk" files with TPHEAD, UVLOD, IMLOD, FITLD, FITTP, and FITAB.. AIPS and some tasks also allow text files to be read or written from/to disk. For example, all print tasks can be instructed to append their output to user-specified text files. These can be examined later with an editor or written to tape with standard tape utilities. The two PostScript tasks, LWPLA and TVCPS, can be instructed to write their output plots in user-specified text files for later processing and, for example, inclusion in manuscripts. And AIPS itself can be instructed to take its input commands from user-created text files.

3.10.1 Disk text files

The most significant user control over external files is the specification of the file's full name, *i.e.*, its directory path and its name in that path. You specify the directory path by creating an environment variable ("logical name" in AIPSpeak) *before* starting AIPS. The simplest way is to change directory (cd Unix utility) to the area you wish to use and enter

% setenv MYAREA 'pwd' C_R

where *MYAREA* is a logical name of your choosing (but all in *upper case*). Note that the pwd is surrounded by backward single quote marks. The grammar above is for users of c-shell and tc-shell. Users of korn, bourne, and bash shells would type:

$ MYAREA='pwd'; export MYAREA C_R

also with backward single quote marks. If you are going to read a text file into AIPS, its name must also be in upper-case letters. Finally, inside AIPS, you specify the file with, *e.g.*,

> OUTPRINT = 'MYAREA:3C123.PRT' C_R

where 3C123.PRT is any all upper-case file name of your choosing. Note the surrounding quote marks and the colon that separates the logical name and the file name portions. You may put the file anywhere under any name you choose, but we request that you put it an area owned by you, if you have one, or that you use an identifying name and a standard AIPS area set aside for the purpose. Files left around in the AIPS directories are subject to summary deletion. Be sure that AIPS has the privilege to write into your directory; use chmod to allow appropriate write privilege on the directory file (try to avoid world write!). On Unix systems, duplicate file names are not allowed and AIPS tasks will usually die when trying to write a file name that already exists. Print tasks will append to pre-existing files, however. The verb FILEZAP allows you to delete external files from inside AIPS. The name of the file to be deleted is given as an immediate argument.

File names may also be entered as complete path names, so long as they do not require more than 48 characters, the length of the adverb data values. Thus

```
INTEXT = '/home/primate2/egreisen/AIPS/Text.prt
```

Note that the trailing quote mark is left off and this is the last command on the input line so that the case is preserved.

Ordinary text files are used in AIPS for a variety of purposes. Every print task offers the option of saving the output in a file specified by OUTPRINT rather than immediately printing and discarding it. Similarly, output PostScript files from LWPLA and TVCPS may be saved in files specified by OUTFILE rather than immediately printing and discarding them. They may be used later in larger displays, or even enclosed as figures in a TeX document such as this CookBook. OUTTEXT is used by numerous other tasks, such as SLICE and IMEAN, to write output specific to the tasks which may be of use to other programs. AIPS tables may even be written as text files by task TBOUT, edited by the user, and then read back in by task TBIN. History files may be revised in a similar manner. Adverbs INFILE, INTEXT, CALIN, and INLIST may be used by a number of tasks to specify source models, lists of "star" positions, holography data, and the like. Television color tables are read from and written to disk text files specified with the OFMFILE adverb.

3.10.2 RUN files

RUN files are ordinary text files containing AIPS commands to be executed in sequence in a batch-like manner. They are often used to define procedures which you save in your own area or in an AIPS-provided public area with the logical name $RUNFIL. The name of the file must be all upper case letters, followed by a period, followed by your user number as a three-digit "extended-hexadecimal" number with leading zeros. (To translate between decimal and extended hexadecimal, use the AIPS procedures or the AIPS verbs called EHEX and REHEX.) The files are edited from Unix level using emacs, vi, textedit or your other preferred text editor. For example, log in to the aips (or your own) account. From Unix level, type:

% cd $RUNFIL   to change to RUN area.

% emacs MAPIT.03D C_R

to edit with emacs a file called MAPIT for user 121. You may now also use any area of your choosing instead of the public $RUNFIL area. For instructions on the individual editors, consult the appropriate Unix Manuals. Instruction manuals for the GNU emacs editor are available from local computer staff. A SAVE area (§ 3.5) may be written as a RUN file if you first GET the area and then use SG2RUN.

To use the RUN file, define a logical name as in the previous Section. Then start up AIPS under your user number and enter

> VERSION = 'MYAREA' C_R   where MYAREA is your disk area, or

\> VERSION = ' ' C_R if $RUNFIL is to be used
\> RUN *FILE* C_R to execute the file named *FILE.uuu*
where *uuu* is your user number if extended hexadecimal with leading zeros to make three digits.

3.10.3 FITS-disk files

FITS is an IAU-endorsed binary format standard for astronomical data heavily used by *AIPS* for almost all of its data on disk and magnetic tape. In fact, it is the only format written by *AIPS* except for simple tape copying. The basic FITS paper (by Wells, Greisen, and Harten) appeared in *Astronomy & Astrophysics Supplement Series*, Volume 44, pages 363–374, 1981. The newsgroup `sci.astro.fits` is devoted to discussion of FITS. World-wide web users can access the FITS home page at

`http://fits.gsfc.nasa.gov/fits_home.html`

AIPS also supports the FITS format written to disk in exactly the same form as it is written to magnetic tape. The tasks `FITTP` and `FITAB` may be instructed to write their output files on disk rather than on tape. Likewise, `TPHEAD`, `FITLD`, `UVLOD`, `IMLOD`, and `PRTTP` can read from disk. To write to a FITS-disk file, specify:

\> DATAOUT '*filename*' C_R where *filename* is the name of the desired output file.

and to read from a FITS-disk file, you specify:

\> DATAIN '*filename*' C_R

where you must specify *filename* with environment variables ("logical names" in *AIPS*peak), *e.g.*,

\> DATAOUT = 'MYDATA:3C123.FIT' C_R

in exactly the same way as described for text files in § 3.10.1. There is a standard public area, called logically `FITS`, which you may use for reading and writing FITS-disk files. `FITTP` will use this area if you do not specify a logical name. Be aware that older files will be purged from this public area when space is needed. Note too that `FITTP` will write only one disk file per execution; the `DOALL` option is disabled when writing to disk.

There is a package of procedures to assist in writing and reading more than one FITS-disk file at a time. Enter `RUN WRTPROCS` to define the procedures. The procedure `FITDISK` will write a single disk catalog file to a disk file using a name based on the *AIPS* file name parameters. You may then construct loops invoking `FITDISK` to write multiple files. For example:

\> FOR I=1:10; GETN(I); FITDISK; END C_R

Such file names are useful for their mnemonic content, but must be read back one at a time. The procedure `WRTDISK` will dump a range of catalog numbers to disk under names that allow the procedure `READISK` to read them back as a group. These two procedures are particularly useful when moving your data between computer architectures (*e.g.*, from a Solaris to a Linux computer). Note that there is a stand-alone program `REBYTE` which can do the conversion directly including files of all types.

`FITLD` can read multiple disk files in either the normal FITS format (as written by `FITTP`) or the special FITS format written by the VLBA correlator. The only requirement for this operation is that file names end in sequential numbers beginning with 1. `FITAB` has the ability to write special FITS files with visibility data in tables. These files may be broken up into multiple files, called "pieces," for size and reliability considerations. These pieces, when written to disk, have names ending in sequential numbers. Special code in `FITLD` and `UVLOD` recognize these pieces and read the requested number of them as if they were in one file.

Remote FITS-disk files may be read in much the same manner as remote magnetic tapes. Type HELP DATAIN C_R or HELP DATAOUT C_R for details.

FITS-disk files are written as Fortran files and hence are available also to user-coded programs. The Fortran specifications for the file are `ACCESS='DIRECT', RECL=2880, FORM='UNFORMATTED'` in the `OPEN` statement for Unix systems. Most Fortrans cannot read or write files larger than 2 Gigabytes, so *AIPS* now reads and writes these files with C subroutines. Users may also, of course, code programs to create such files to be read by `FITLD`, `IMLOD` or `UVLOD`. Consult *GOING AIPS*, Volume 2, Chapter 13 for details on how to do this.

One of the main uses for FITS-disk files is to transfer data over the Internet between computers. For example, to transfer a file from `rhesus` (in Charlottesville) to `kiowa` (at the AOC), log in to `rhesus`, change to the directory in which you wish to store the file (for example, cd $FITS C_R), and enter:

% ftp kiowa C_R	to start `ftp` to the remote system.
`Name (kiowa:...):` *loginame* C_R	to log in to account *loginame*.
`Password:` *password* C_R	to give the account's password.
`ftp>` cd *directory* C_R	to change to the *directory* name containing the file.
`ftp>` binary C_R	to allow reading of a binary file.
`ftp>` hash C_R	to get progress symbols as the copy proceeds.
`ftp>` put *filename* C_R	to send the file
`ftp>` quit C_R	to exit from `ftp`.

The file should then be in the desired directory. You may have to rename it, however, to a name in all upper-case letters since that may be required by *AIPS*. (See § 3.10.1 for a trick that allows you to use lower-case letters in file names.) The file format will be correct. In general it is better to use the ftp program to "get" files instead of "put"ting them; things tend to go faster that way.

An alternative to using `ftp` is to use the `rcp` (remote copy) Unix utility or to write the output file directory in the appropriate area on the other computer. In order to do this, you have to have accounts on both machines, and you should have set up a `.rhosts` file (see the Unix manual page on `rhosts` for instructions). Once you know this works (test it via, *e.g.*, `rsh rhesus whoami`), the syntax for the remote copy is:

% rcp $FITS/MYFILE.FITS kiowa:/AIPS/FITS/MYFILE C_R

(this shows how you would copy it from `rhesus` to `kiowa`). A secure copy (`scp`) would be better if you have set up the secure connection capability.

If you wish to copy a FITS-disk file from one machine to another within a site, check if you can just use the Unix `cp` command; this is often possible if the remote disk is mounted (or can be auto-mounted) via NFS (the Network File System).

FITS files may be compressed with standard utility programs such as `gzip`. This does not produce much compression for files written with full dynamic range and floating-point format. However, `FITAB` offers the option of writing images (not *uv* data) which are quantized at some suitable level. These are capable of significant compression even if they are in floating-point format.

3.10.4 Other binary data disk files

Data written by the on-line system of the VLA are now often found in disk files rather than on tape. These data are available from an archive of all VLA data. See

`https://data.nrao.edu/`

for information on how to access your current data and all data for which the proprietary period has expired. `FILLM` and `PRTTP` can read the disk files produced from the archive, including reading more than one such file in a single execution. In this case, the file names must end in consecutive numbers beginning with `NFILES` $+ 1$.

3.11 The array processor

In running numerous important tasks in *AIPS* you will notice references to an "array processor." This used to be an expensive device attached to computers which allowed them to run some *AIPS* tasks 100 times faster than they would run without the device. To support our less fortunate colleagues, we wrote a software emulation of the array processor which we call the pseudo array processor. This uses highly optimized routines running on data stored in the "AP memory." By now, the hardware devices are all gone and only the software emulation remains. We have found that this is still a good model to obtain highly optimized software performance.

The pseudo-AP now uses dynamic memory allocated by the calling task as needed rather than a fixed amount of memory that would be much too large for some problems and too small for others. The largest *AIPS* tasks, including `IMAGR` and `CALIB` in self-cal mode, have changed to use rather large amounts of memory if needed to reduce the number of times the visibility data need to be read. Two new verbs appeared at this time: `SIZEFILE` returns the size of a disk file and `SETMAXAP` sets the maximum computer memory that may be used for the pseudo-AP. The latter allows the user to limit the AP on smaller or busier machines or to permit really large memory usage when large amounts of memory are available. See HELP SETMAXAP for a discussion of the essential considerations.

The pseudo-AP has been changed to use only double precision integers and floats. This avoids serious overflow issues with the large number of data samples from modern telescopes and provides much greater accuracy especially in the gridding and Fourier transforming of visibility data.

3.12 Additional recipes

3.12.1 Columbian fresh banana cake with sea foam frosting

1. Open an 18.5 oz **yellow cake mix** into a large mixxing bowl; do not use a mix that contains pudding or requires oil. Combine with 1/8 teaspoon **baking soda**.
2. Stir 3/4 cup **Coca-Cola** briskly until foaming stops. Add to cake mix and blend until just moistened. Then beat at high speed for 3 minutes, scraping the bowl often.
3. Combine 2 teaspoons **lemon juice** with 1 cup mashed **bananas** and then add to batter. Add 1/3 cup finely chopped **nuts** and beat for 1 minute at medium speed.
4. Turn the batter into a well greased, lightly floured 9x13 baking dish. Bake in a preheated 350° F oven for about 40 minutes or until the cake tests done. Cool on a rack for 15 minutes, remove cake from pan and turn right side up on a rack to finish cooling.
5. In the top of a double boiler, combine 2 large **egg whites**, 1 1/2 cups packed **light brown sugar**, 1/8 teaspoon **cream of tartar** (or 1 tablespoon corn syrup), and 1/3 cup **Coca-Cola**. Beat at high speed for 1 minute with an electric mixer.
6. Place over boiling water — the water should not touch the bottom of the top half of the double boiler. Beat on high speed for about 7 minutes until the frosting forms peaks when the mixer is raised. Remove from boiling water. Empty into a large bowl.
7. Add 1 teaspoon **vanilla extract** and continue beating on high speed until thick enough to spread, about 2 minutes. Spread on the sides and top of the cold banana cake.

3.12.2 Banana sweet potato puff casserole

1. In a large bowl, combine 2 cups mashed **sweet potatoes**, 1 cup mashed ripe **bananas** (3 medium), 3/4 teaspoon **curry powder**, 1/3 cup **sour cream**, 1/2 teaspoon **salt**, and 1 **egg**.
2. Beat with electric mixer until light and very fluffy. Turn into 1 quart casserole dish.
3. Bake at 350° F for 20 minutes or until puffed and lightly browned.

Thanks to Turbana Corporation (`www.turbana.com`).

3.12.3 Banana poundcake

1. Mix in large bowl until blended:
   - $1\frac{1}{3}$ cups mashed **bananas** (4 medium)
   - 1 pkg. ($18\frac{1}{2}$ oz.) **yellow cake mix**
   - 1 pkg. ($3\frac{3}{4}$ oz.) instant **vanilla pudding mix**
   - $\frac{1}{3}$ cup **salad oil**
   - $\frac{1}{2}$ cup **water**
   - $\frac{1}{2}$ teaspoon **cinnamon**
   - $\frac{1}{2}$ teaspoon **nutmeg**
   - 4 **eggs** at room temperature
2. Beat at medium speed for 4 minutes.
3. Turn batter into greased and lightly floured 10-inch tube pan.
4. Bake in 350° F oven for 1 hour or until cake tester inserted in cake comes out clean.
5. Cool in pan 10 minutes, then turn out onto rack and cool completely.
6. If desired, dust with confectioners sugar before serving.

Thanks to the United Fresh Fruit and Vegetable Association.

3.12.4 Banana July cocktail

1. Sprinkle 3 sliced **bananas** with 1 tablespoon **lemon juice**.
2. Mix with 1 1/4 cans drained and flaked **tuna**, 1/2 **onion** chopped, and 2 tablespoons chopped **gherkins** or **olives**.
3. Spoon into 7 cocktail shells.
4. Melt 2 tablespoons **butter** in a saucepan. Add 2 tablespoon **cake flour** and salt and pepper to taste.
5. Add 1/4 cup **chicken stock** and 1/4 cup dry **white wine**. Simmer for one minute stirring constantly.
6. Add 1/3 cup grated **cheddar cheese** and allow to cool.
7. Add 1/4 cup fresh cream to sauce and pour over banana-tuna mixture.
8. Sprinkle with 1 tablespoon grated **cheese** and **paprika**. Decorate with a slice of **gherkin** or **olive**.
9. Bake 15–20 minutes at 350° F; serve warm.

Thanks to Turbana Corporation (`www.turbana.com`).