AIPS at the Australia Telescope National Facility

Mark Calabretta

Australia Telescope National Facility
CSIRO, PO Box 76, Epping NSW 2121, AUSTRALIA
email: aipsmngr@atnf.csiro.au

August 8, 1991

1. Introduction

The Australia Telescope National Facility (ATNF) maintains a version of AIPS with many local modifications. Most of these are general enhancements, although some are designed specifically for reduction of AT data.

AIPS has been in continual use by the ATNF, or its progenitor the CSIRO Division of Radiophysics, since about 1983. Until 1988, the implementation was exclusively VMS with very limited networking and effectively only the one Epping site. Since the acquisition of a Convex in 1988, we have migrated to unix, and built up a large network which includes four separate sites.

This memo describes some of the mechanisms which have been developed to help manage AIPS at ATNF. In doing so it addresses some of the problems which have been raised in the past by other AIPS managers attempting to adapt AIPS to their local systems.

2. Hardware

The ATNF has four main sites: the headquarters located in Epping, a Sydney suburb, the Compact Array located near Narrabri about 550 km NNW of Sydney, the Mopra telescope near Coonabarabran about 450 km NW of Sydney, and the 64m Parkes Telescope located about 300 km west of Sydney.

The ATNF configuration of AIPS is predicated in part by the type and distribution of hardware. The system, which is in a constant state of flux, may be summarized as follows:

Epping:

* A Convex C220 with 128 Mbyte memory, two 9-track tape drives, 550 Mbyte of disk space for the AIPS system, and 4 Gbyte for AIPS data.

* 15 Sun SPARC workstations for production AIPS
  - SLC (x2, diskless)
  - IPC (x7, each with an internal 207 Mbyte disk, one has two 1 Gbyte AIPS data disks plus Exabyte tape drive)
  - SPARC 1 (x4, each with an internal 105 Mbyte disk, one has a 1 Gbyte AIPS data disk plus Exabyte tape drive)
  - SPARC 2 (x1, one internal 207 Mbyte disk drive, two 1 Gbyte AIPS data disks, two Exabyte tape drives)
  - 4/370 (x1, with local disk and TAAC graphics accelerator)

* 1 Tektronix X-terminal

* A number of other Sun SPARC workstations owned by Radiophysics which are used for AIPS TV display but don't run AIPS itself.

Narrabri:

* 3 Sun SPARC workstations for production AIPS
  - SPARC 2 (x1, one internal 207 Mbyte disk drive, one 327 Mbyte disk for the AIPS system, two 1 Gbyte AIPS data disks, Exabyte tape drive)

SPARC 1+ (x2, each with an internal 105 Mbyte disk,
each has two 669 Mbyte AIPS data disks,
one has an Exabyte tape drive)

Parkes:
~~~~~~
* 1 Sun SPARC workstation for production AIPS
SPARC 1+ (x1, two internal 105 Mbyte disks,
one 327 Mbyte disk for the AIPS system,
one 669 Mbyte AIPS data disk,
one Exabyte tape drive)

Mopra:
~~~~~
* A SPARC 2 will be purchased with several Gbytes of disk space.

Apart from the SLC's, each workstation has at least one disk drive for local root and swap. More use was originally made of diskless clients but this placed too great a load on the server and network, and the strategy now is only to serve /usr. In SUN's terminology they are "dataless" clients.

3. The LOCAL directory heirarchy

The ATNF is active in the area of AIPS programming and various tools have been developed to facilitate the evolution of our system in parallel with that of NRAO.

The golden rule of ATNF AIPS code management is that the copy of AIPS handed down from on high in Charlottesville shall remain untouched unto the last bit - at least until the next release! Experience has shown that the installation of a new release of AIPS is that much more tedious and complicated when additions or modifications reside in the NRAO given AIPS areas.

In practice, the NRAO areas are strictly read-only. Since the intermediate products of compilation are sent to the $PREP area (described below) write-permission is not needed even for that purpose. Instead, a parallel directory heirarchy has been created to accomodate new and changed code. This is rooted at the same level as $AIPS_VERSION, so for example, if

$AIPS_VERSION = $AIPS_ROOT/15APR91

then

$LOCAL = ${AIPS_VERSION}_LOCAL = $AIPS_ROOT/15APR91_LOCAL.

A set of LOCAL directory environment variables shadows that of the NRAO directories. These are formed from the standard name with "LOC" prefixed. For example, if

$AIPPGM = $AIPS_VERSION/AIPS/PGM,

then

$LOCAIPPGM = $LOCAL/AIPS/PGM

and so on for all code directories used locally.

The LOCAL tree is implemented in unix by modifying the standard AIPS utilities, particularly SEARCH and MAKEAT so that they look for code in the LOCAL area before considering the NRAO area. LIBR.DAT also defines libraries and link lists for local subroutines and tasks. In VMS, the system in which the LOCAL tree was originally constructed, the LOCAL tree was implemented via the elegant simplicity of iteratively translated logical name search paths.

AIPS source code which has been modified locally is processed by a utility called ENLOC. This is built on top of the unix "diff" utility and identifies modified sections of code, tagging them with the date and identity of the programmer who made the change. It has proved to be indispensible, especially when a new release of AIPS is installed and all local modifications have to be reconciled with changes made in Charlottesville. The following code fragment illustrates an ENLOC difference section and comes from our version of ZPHFIL.FOR which has been modified to allow up to 255 TV device numbers (TVs

are discussed later)

```
*>                                          MRC 90/Nov/15:
        pnam = 'TVDEV  :'
        call zhex (ntvdev, 2, pnam(6:7))
*=
*        WRITE (PNAM,1020) NTVDEV
*<
```

In FORTRAN the locally modified section is cast to lower case to distinguish it from NRAO source. ENLOC recognizes FORTRAN (including *.INC), C (including *.H), AIPS help files, VMS procedure files, Bourne Shell and C Shell scripts. It is applied automatically by COMRPL and COMLNK so there is little chance of a code modification slipping past its notice.

A complimentary utility, DELOC, undoes ENLOC difference sections in one of three ways: 1) removes the NRAO half of the difference section and casts the local section to upper case, but leaves the ident line intact, 2) as for 1, but also removes the ident line, 3) reconstructs the original NRAO code. In recent times a large quantity of ATNF AIPS code has been exported back to Charlottesville (particularly by Eric Greisen who has been on sabbatical here for the past 18 months), and option 2 is intended to facilitate this.

Inevitably, the ATNF AIPS modifications exported to Charlottesville have returned to us in subsequent releases of AIPS. To account for this an "install" option was added to ENLOC when we installed the 15APR91 release. It successfully caught our exported modifications and rationalized the difference sections. In many cases no additional changes had been made in Charlottesville, and with no residual difference sections, ENLOC simply deleted the LOCAL code. A recursive "diff" on the 15JUL90_LOCAL and 15APR91 directory trees had produced a list of differences 110,000 lines long. By dividing this among four programmers, and using the ENLOC install option, the rationalization was completed in about 2 days.

4. AIPS management in a distributed system

In addition to the four ATNF sites described above, we currently provide Exabyte "tar" copies of ATNF AIPS to other Australian institutions for reducing AT data (they are registered with NRAO). From the software management point of view, this is considerably simplified by structuring our AIPS system so that only one copy is required at any site (effectively local area network). Moreover, the master copy, which resides on the Convex at Epping, can simply be copied to other sites and run with minimal setup requirements.

The main features of the implementation are:

* all machines at a site access a single copy of the local AIPS system

* the AIPS directories have been restructured to support multiple hosts and architectures without duplication of source code

* csh "source" scripts ascertain the local site, host, architecture, and AIPS environment, and set relevant environment variables such as $PATH accordingly.

One machine at each site is designated as AIPS master and provides the AIPS system to all clients of whatever architecture. The service has been implemented exclusively in the unix domain via Network File System (NFS), for Convex, SUN (both Sun 3, and Sun SPARC), and IBM machines. However, it originated with a pair of VAXes sharing a common disk farm.

Modification of the AIPS directory tree involved introducing architecture- and host-specific subdirectories. Currently supported architectures ($ARCH) are CVEX and SUN4, but in the past we've also had SUN3, and IBM. Instead of having a single $AIPS_VERSION/LOAD area we have a $AIPS_VERSION/$ARCH/LOAD for each architecture. For example, on the Convex $ARCH translates to "CVEX", whence $LOAD = /AIPS/15APR91/CVEX/LOAD. The following directories are architecture specific:

```
$LOAD       $AIPS_VERSION/$ARCH/LOAD
$LOADn      $AIPS_VERSION/$ARCH/LOAD/ALTn
$LIBR       $AIPS_VERSION/$ARCH/LIBR
```

```
$PREP        $AIPS_VERSION/$ARCH/PREP
$SYSLOCAL    $LOCSYSUNIX/$ARCH
$LOCINSUNIX  $LOCSYSUNIX/INSTALL/$ARCH
```

The $PREP area is provided to hold preprocessed code and object modules for each architecture. The basic tenet is that the code areas must be used for code AND NOTHING ELSE. Object modules created by machines of one architecture must not be visible to machines of any other! The AIPS utilities PP, COMRPL, COMLNK, AS, CC, FC, and LINK have all been modified to send their intermediate output to $PREP. To save disk space, the intermediate products of compilation (.f, .c, .s, .o, .LOG) are not retained as they are in standard AIPS unless the NOPURGE option is specified to COMRPL or COMLNK.

The following directories are host specific:

```
$MEMORY      $AIPS_VERSION/MEMORY/$HOST
$TSTMEM      $TST/MEMORY/$HOST
$NEWMEM      $NEW/MEMORY/$HOST
$OLDMEM      $OLD/MEMORY/$HOST
$ERRORS      $AIPS_VERSION/ERRORS/$HOST
```

The AIPS system directory is a special case, the AC (accounting), BA (batch job), BQ (batch queue), SP (system parameter), TC (task communication), TD (task data), and TP (tape lock) files are host-specific. On the other hand, the GR (gripe), IC (image catalogue), ID (image device), and PW (password) files are site-specific. The solution is to have host-specific directories

```
$DA00        $AIPS_ROOT/DA00/$HOST
```

and take care of the site-specific files by creating hard links between them in each directory. Thus, for example, users may change their AIPS password while running AIPS on one machine and the change will be effected on all others. The TV catalogue and device files will be discussed more fully later.

Creation of the MEMORY, ERRORS, and DA00 directories and the plethora of symbolic links therein would be a tedious operation if done by hand even for a modest number of hosts, and so a system utility (SYSETUP) has been written to do it automatically. Likewise, changing system parameters for a collection of hosts via SETPAR is also very tedious, and task SETSP has been written to make changes collectively.

Definition of the fundamental SITE, HOST, and ARCH environment variables is handled by a csh "source" script called $AIPS_ROOT/HOSTS.CSH, a sample copy of which is appended at the end.

Once $ARCH is known the AIPS path may be defined, and this is handled by $AIPS_ROOT/AIPSPATH.CSH. This csh source script replaces some of the functionality of the standard LOGIN.CSH script, doing as much as is necessary to redefine the path. It first defines the AIPS versions, OLD, NEW, and TST and then sets AIPS_VERSION to $NEW by default or to the value indicated by $VERSION if defined. It then defines the architecture-specific system areas and finally the PATH. A sample copy of AIPSPATH.CSH is also appended below.

In standard AIPS, the source script that defines AIPS devices is $SYSLOCAL/ASSNLOCAL.CSH. Its more cosmopolitan replacement in ATNF AIPS is $AIPS_ROOT/AIPSASSN.CSH, which provides for device assignments peculiar to a particular SITE or HOST. A sample copy is appended.

Tape allocation in ATNF AIPS differs somewhat from standard. AIPS accesses tape drives through the MT0n logicals, and these are initially set by AIPSASSN.CSH to the rubbish value "UNMOUNTED". Any attempt by an AIPS user to access an unmounted tape therefore results in an error. When tape "n" is mounted, ZMOUN2.C sets MT0n to the name of the raw no-rewind device with which it is associated. The CVEX version of ZMOUN2.C uses hard-coded device names as in standard AIPS, but the symbolic link is created in $DA00 instead of $AIPS_ROOT. On the SUNs, ZMOUN2.C ascertains the tape device by translating logical TAPEn, and this is set on a host-specific basis by AIPSASSN.CSH. Since SunOS has no tape allocation facility, ZMOUN2.C also attempts to gain exclusive use of the tape drive by creating a lock file, $DA00/TAPEn.lock, which serves as a mount lock among cooperating AIPS processes. By contrast, the TP system file used by ZTPOPN.FOR as an access lock only has currency while the tape device is actually open. The $APLSUN tape routines have been modified as necessary to allow Exabyte usage.

5. AIPS users

For security reasons, and also for accounting purposes, the ATNF does not allow generic accounts such as "guest", "visitor", or "aips". AIPS users must therefore run AIPS from their own account. For added security, ATNF AIPS is structured so that only users in the "aipsusr" group have permission to run AIPS or access or modify AIPS data (note that, unlike VMS, a unix user may belong to as many as eight groups, and this is done by adding the user to the "aipsusr" group in /etc/group).

With users coming and going on a regular basis, it is clearly desirable that they be able to run AIPS without special modifications to their .login or .cshrc files. The only assumption made is that users have /usr/local/bin in their path. They may then start AIPS via /usr/local/bin/aips which is a symbolic link to the generic startup script $AIPS_ROOT/START_AIPS.

START_AIPS performs many of the functions of the standard startup script, $SYSUNIX/AIPS. It is a separate csh script since it needs to source HOSTS.CSH, AIPSPATH.CSH, AIPSASSN.CSH, and certain others (TVDEVS.CSH and DADEVS.CSH, see below) which define essential environment variables. (The dichotomy between csh "source" scripts and Bourne shell "." scripts has proved to be a very irritating flaw in the design of csh. The implementation of the AIPS scripts as csh source scripts was predicated by the general use of csh as the default interactive shell.)

START_AIPS parses user options and, in particular, sets $VERSION on the basis of an OLD, NEW, or TST option before sourcing AIPSPATH.CSH. This means that AIPS_VERSION is only ever defined or redefined by version-independent scripts. The original system of having the version-specific AIPS script redefine AIPS_VERSION is illogical.

6. TV devices

ATNF AIPS considers that TV display devices are a network resource. The design goal was to allow a user on any machine to specify any display device by name as a command line option to START_AIPS. If no display device is specified, and START_AIPS determines that the user is sitting in front of a recognized AIPS display workstation, it will be used by default (this can be defeated by using a "NOTV" option). Part of the implementation involves determining what windowing system the workstation is using (e.g. SunView or X-Windows) and starting the appropriate screen server. X-terminals are also catered for. ATNF AIPS also runs a Tektronix graphics server, TEKSERVER, which is not available in the standard release.

An AIPS user using a workstation with multiple windows may be running AIPS on one machine in one window, and on a different machine in another. In order to coordinate TV usage, AIPS on either machine must therefore access a common image device (ID) file (used for locking purposes). They must also share the same image catalogue (IC file). These files reside in the AIPS system directory ($DA00) which is host-specific, but as explained above, this conflict is resolved by using hard links between the IC and ID files in each directory.

The script $AIPS_ROOT/TVDEVS.CSH coordinates network usage of all AIPS display servers at a particular site. It sets a (hard-coded) environment variable $TVS to list the names of all recognized servers, and this is used for argument parsing by START_AIPS. The order of the names in the list also determines the AIPS TV device numbers. If $TV has been defined by START_AIPS and translates to the name of a recognized server, TVDEVS.CSH will then set the environment variables required by AIPS to access the server. These are

TVALT	...defines the display server to use (SSS, XAS, XVSS)
TVDEV	...defines the AIPS TV display number
TVDEVxx	...defines the display service
TKDEV	...defines the AIPS graphics device number
TKDEVxx	...defines the graphics display server

A server may be specified in the form "display:server" in the list of TVS to indicate that an X-server has a display other than itself, for example an X-terminal. Since TVs are a network resource, screen servers always use

Internet domain sockets.

In many cases TVDEVS.CSH determines the TVALT environment variable by polling the server itself via

```
setenv TVALT 'rsh $TV -n $AIPS_ROOT/TVALT'
```

The TVALT script simply looks for the "sunview" or "olwm" processes to decide which windowing system is running. Differing screen server characteristics are handled by assigning different AIPS device numbers for each server implementation. For example, a server running SSS as AIPS TV number 5 will be assigned device number 105 when it's running XAS. The image device (ID) file for TV number 5 would be configured for SSS, and that of TV number 105 for XAS. However, to save disk space the image catalogue (IC) files for device 105 are linked symbolically to those of device 5. AIPS was modified as necessary to allow for more than 15 image devices. We have more than 15 anyway.

The TV servers themselves are initiated by $AIPS_ROOT/START_SERVERS which is activated on the server machine via

```
rsh $TV -n $AIPS_ROOT/START_SERVERS &
```

from the client. START_SERVERS is activated from the Bourne shell AIPS script rather than START_AIPS in order to circumvent the tedious and unavoidable process messages produced by csh. START_SERVERS establishes the server's identity by sourcing HOSTS.CSH, TVDEVS.CSH, and AIPSPATH.CSH, the latter of which allows it to find the correct executables for the server's architecture. Depending on the value of TVALT, START_SERVERS actually calls one of SSSERVERS, XASERVERS, XVSERVERS, or XTSERVERS to do the dirty work. These shell scripts reside in $SYSLOCAL since they may be architecture-specific.

The TV implementation described here does not use TVMON, the AIPS TV-by-wire server, simply because it is unnecessary and would add an extra network overhead. It also assumes that TVs are directly addressable via the network. This is obviously not the case for older style TVs such as the IIS, or DEANZA. However, these host-bound peripherals could readily be implemented by using TVMON.

It should also be noted that the graphics implementation is incomplete in that the dichotomy between TEKSERVER and REMOTE graphics terminals has not been resolved. The TKDEV graphics device number currently assigned to TEKSERVER for a particular workstation is the same as the SSS TV device number. However, the AIPS model assumes that there will be no more than 10 dedicated Tektronix terminals and ZWHOMI.FOR assigns TKDEV device numbers for REMOTE terminals as 10+NPOPS. These numbers may overlap with that of the TEKSERVER graphics terminals, but worse, since the graphics catalogue (ICC00000;1) is now shared among all hosts by virtue of the hard links, conflicts could arise between REMOTE terminals running on different hosts with the same POPS number. One solution would be to have separate network and host-private graphics catalogues. In practice, we run REMOTE graphics terminals so rarely that there has never been a conflict, so there has been no incentive to fix the problem.

7. Network data disks

With the advent of NFS and IEEE standard floating point format, ATNF AIPS considers that AIPS data disks are a network resource. The design goal was to allow an AIPS user on any machine to access the data disks on any other by specifying them by name as a command line option to START_AIPS. If no disks are specified, the user will be allocated the data disks from all hosts that have been declared as "required", plus any local disks.

Network data disks are implemented by $AIPS_ROOT/DADEVS.CSH. This receives as input an environment variable $DAOPT, being a colon-separated list of AIPS hosts whose disks are to be allocated. It contains a (hard-coded) list of all available AIPS data disks, the order of which is preserved when AIPS disk numbers are assigned. Disks have names of the form "HOST_x", where HOST is the name of the host to which the disk is attached, and "x" is a single digit between 1 and 9. DADEVS.CSH defines the disk environment variables, $DA0n, required by AIPS for each data area, and also $NVOL, the number of disks allocated. AIPS disk 1 is special since it is the repository for each user's message and save/get files, so it should be the first in the list and always "required".

DADEVS.CSH tests the validity of each AIPS data area by testing for the existence of a file called "PRESERVE". This ensures that NFS mounted data areas are in fact mounted. The PRESERVE file is actually an ASCII text file containing a list of users who are to be spared from the ravages of TIMDEST. One drawback of the network disk system is that the AIPS number of a data disk may change from one invocation to another. However, users can determine the correspondence between the AIPS disk number and the actual device within AIPS via verb FREESPAC which has been modified to list data areas in the form "HOST_x".

ZDCHIN, the subroutine which fills the device characteristics common, has been modified to test whether the $NVOL environment variable is defined. If so, its translated value overrides the number of disks declared in the host's system parameter (SP) file and ZDCHIN is caused to read the network system parameter file, $NET0/NETSP. NETSP is an ASCII text file containing a one-line entry for each disk, and overrides the TIMDEST and disk reservation information in the SP file.

It might seem simpler for each AIPS host always to allocate all available AIPS disks on the network. However, this is inadvisable since processes on an NFS client hang if they try to access a disk from a server which has crashed. Since AIPS routinely accesses the catalogue file on all disks known to it, this is a strong disincentive against allocating data areas in which the user has no interest. Furthermore, AIPS only supports 15 simultaneous data disks (probably as many as a user could handle at one time) and we already exceed that.

One of the principle reasons for the development of the network disk system was simply the relatively greater expense of procuring peripherals for the Convex compared to the workstations. If one stops to consider the flow of data out of the AT and through the data reduction phase, it is clear that a bottle-neck will be formed if the available processing power is not sufficient to handle the data as fast as it is acquired. However, since the first stage of the data reduction consists of editing the uv data interactively, it is apparent that a large amount of cheap, but not particularly fast, disk space is required. Furthermore, editing and calibration are not very CPU-intensive, and are certainly well within the capabilities of the workstations. Assigning these tasks to them has the consequent benefit of offloading some of the work from the Convex.

In our experience, disk space can be procured for a workstation at rather less than one quarter of the cost of that for the Convex. Naturally it doesn't bear comparison in terms of speed, but speed is not of the essence for uv editing or calibration. On the other hand, Exabyte drives for the Convex are at least a factor of eight more expensive than for a workstation since they are only supplied by Convex as a "special". This effectively stimied our plan to put Exabytes on the Convex, and left us in a quandry since the AT has standardized on Exabytes for data archiving and transport. Before we acquired data disks on the workstations we routinely used the Exabytes on the workstations to write to the Convex's disks which were NFS mounted, and this was painfully slow.

Now, it must be realized that NFS write is about an order of magnitude slower than NFS read (yes, that's right, a factor of 10). Therefore, at least for large data sets, one should organize things so that data is only ever read from an NFS file system, never written. The current strategy for reducing AT data is

1) on a workstation with local data disks and Exabyte drive, load the data (ATLOD, FILLM, UVLOD) from the local Exabyte to the local disk

2) display (LISTR, PRTUV, UVPLT), edit (TVFLG, SPFLG, UVFLG, CLIP, QUACK), and calibrate (SETJY, CALIB, SNPLT, GETJY, CLCAL, BPASS) the uv data with AIPS running on the workstation

3) then with AIPS running on the Convex, run SPLIT so that it reads from the workstation's disk and writes to one of its own disks

4) start number crunching (MX, etc.) on the number cruncher

This system is still in its infancy but the early signs are all positive. Probably its main difficulty is user education.

Another AIPS management utility, HOGS, deserves a brief mention. It simply prepares a summary of the major AIPS disk users from information collected at

midnight each night by the unix "quot" accounting utility. The summary is listed by GRUNT (described below) when a user starts AIPS. Before HOGS runs each night, the DAOWN script resets the ownership of all AIPS data files to correspond with the user information stored in the $AIPS_ROOT/AIPSUSERS database.

A number of disks on the Convex and workstations are managed by the local booking system. These may be booked for up to a week, thus ensuring that enough disk space is available for visitors and locals, particularly spectral line users, who have large data cubes to process. Bookings for each disk are recorded (by a human moderator) in an ASCII text file. These are processed each night by a "cron" job which updates the $NET0/NETSP file with the AIPS user numbers of those granted access. Only booked users "see" these disks from within AIPS by virtue of AIPS' inbuilt disk reservation system. The booking cron job optionally deletes the files of users whose bookings have expired. It also produces a booking summary to be displayed by GRUNT.

8. Debugging

The debugging strategy adopted in ATNF AIPS is oriented towards conserving disk space. Task executables are generally stripped of their symbol tables, but debugging information may be regenerated easily when required. To this end, subroutines are always compiled and maintained in their object libraries with the debugger symbol table attached. The Convex FORTRAN compiler allows this at any level of optimization, although debugging optimized code can be a nightmare. On the other hand, the SUN compiler only allows debugging of unoptimized binaries and we currently accept this limitation, particularly since we are unsure of the reliability of the optimization. However, as shown in AIPS memo 67, the SUN FORTRAN compiler provides a speed increase of as much as a factor of x2 at the highest level of optimization. If at some future time we decide we need this, we will probably opt to keep separate libraries for optimized and unoptimized object modules.

The symbol table accounts for a significant fraction of the size of an executable compiled in debug mode. The disk space that this would consume is saved by always maintaining the AIPS task executables with their symbol tables stripped off. Unless the DEBUG option has been specified to COMLNK, the LINK procedure automatically passes the "-s" option to "ld" to explicitly strip off the symbol tables inherited by the executable from those attached to the subroutine object modules. (The same effect can be achieved with the unix "strip" utility.) To debug a task, it is necessary to use COMLNK with the DEBUG and NOPURGE options. Any subroutines of interest also have to be preprocessed by running PP manually since preprocessed code is not normally retained.

9. GRUNT

GRUNT is a simple but very useful mechanism for sending messages to AIPS users. Messages are stored, one per file, in a site-specific directory, $MSGS = $AIPS_ROOT/SITES/$SITE/MSGS, with five character names of the form [IHBUM][0-9][0-9][0-9][+-], for example U081+. The initial letter defines a message class, and the following three digits a sequence number. A trailing "+" indicates that a message is current. When a message becomes stale the "+" is changed to "-" thereby deactivating it, but still preserving it for historical purposes.

GRUNT is activated when AIPS starts up via a command of the form

```
GRUNT $MSGS/[IUHB]*+
```

GRUNT prints any current messages which the user has not already read, asking after each "Have you finished reading this message?", the default answer being no. If the user acknowledges having read the message, GRUNT appends the user's name to the end of the message in a form which distinguishes it from the message text. The user will not receive the message again, but can see all current messages by using the "news" command which invokes GRUNT in news mode. GRUNT has the added benefit of protecting AIPS programmers from users who ignore messages but claim they "were never told". Aborting GRUNT aborts the

AIPS startup. Message classes currently implemented (in list order) are

```
I: informational messages for new users
U: general user messages
M: AIPS management messages (only for programmers)
H: disk usage figures produced by HOGS
B: current disk bookings
```

In order to protect first-time users from an avalanche of information, GRUNT gives the user the option of deferring the fourth and subsequent messages. GRUNT runs in setgid mode so that it can append to the message files which have "aipsprog" group ownership.

10. Implementation summary

The following scripts of local origin are appended:

```
$AIPS_ROOT/HOSTS.CSH
$AIPS_ROOT/AIPSPATH.CSH
$AIPS_ROOT/AIPSASSN.CSH
$AIPS_ROOT/START_AIPS
$AIPS_ROOT/TVDEVS.CSH
$AIPS_ROOT/TVALT
$AIPS_ROOT/START_SERVERS
$SYSLOCAL/SSSERVERS
$SYSLOCAL/XVSERVERS
$SYSLOCAL/XASERVERS
$SYSLOCAL/XTSERVERS
$AIPS_ROOT/DADEVS.CSH
$NET0/NETSP
```

(All references to non-ATNF sites have been removed from the versions appended.)

A brief summary of changes made to NRAO (unix) scripts to implement the features described above:

```
AIPS          ...activate START_SERVERS
              ...activate GRUNT
              ...defeat all TV device assignments
              ...defeat all attempts to reset AIPS_VERSION
              ...use $LOAD instead of $AIPS_VERSION/LOAD
AREAS.CSH     ...define LOCAL directories explicitly
              ...define architecture- and host-specific areas
AS            ...send intermediate output to $PREP
CC            ...send intermediate output to $PREP
CDVER.CSH     ...use $AIPS_ROOT/AIPSPATH.CSH
COMLNK        ...send intermediate output to $PREP
              ...apply ENLOC
              ...always delete the object module
COMRPL        ...set default to DEBUG
              ...send intermediate output to $PREP
              ...apply ENLOC
FC            ...send intermediate output to $PREP
INCS.SH       ...add LOCAL include directories to the search list
LDOPTS.SH     ...remove $DEBUG as default and add the "-s" option in its place
LIBR.DAT      ...define libraries for LOCAL subroutines, and link lists
                 for LOCAL tasks
LINK          ...set "-s" option to strip the symbol table for NODEBUG opt
              ...send intermediate output to $PREP
MAKEAT        ...search LOCAL directories before NRAO
PP            ...send intermediate output to $PREP
PP.FOR        ...allow extra characters in the translation of include
                 directory environment variables
RUN           ...use $LOAD instead of $AIPS_VERSION/LOAD
SEARCH        ...search LOCAL directories before NRAO
              ...ignore any .f, .c, .s, or .o files
```

Modifications to AIPS code

```
AU3A.FOR      ...modify TIMDEST to use PRESERVE file
VERMAT.FOR    ...look for help files in $LOCHLPFIL first
```

```
   YTVOPN.FOR  ...a FORMAT statement had to be modified to allow up to 255 TV
                  devices
   ZACTV8.FOR  ...look for executables in $AIPS_VERSION/$ARCH/LOAD
   ZCRLOG.C    ...fixed for general case (required for tape system)
   ZDCHIN.FOR  ...implement network data disks
   ZFRE2.C     ...list disks by host name and number
   ZPHFIL.FOR  ...allow up to 255 TV and graphics devices
   ZTKCL2.C    ...implement TEKSERVER
   ZTKOP2.C    ...implement TEKSERVER
   ZWHOMI.FOR  ...allow up to 255 TV and graphics devices

   CVEX/ZMOUN2.C    ...put tape symbolic link in $DA00
   CVEX/ZSTAI2.C    ...put tape symbolic link in $DA00

   SUN/ZBKLD2.FOR   ...Exabyte modifications
   SUN/ZBKTP1.FOR   ...Exabyte modifications
   SUN/ZBKTP2.FOR   ...Exabyte modifications
   SUN/ZMOUN2.C     ...implement tape system
   SUN/ZTAP2.C      ...Exabyte modifications
   SUN/ZTAPE.C      ...Exabyte modifications
   SUN/ZTPCL2.C     ...Exabyte modifications

The following NRAO scripts are defunct in ATNF AIPS:

   LOGIN.CSH
   ASSNLOCAL.CSH
   BATER

Addenda:
>>>>>>>>>>>>>>>>>>>>>>>>>>>> $AIPS_ROOT/ HOSTS.CSH <<<<<<<<<<<<<<<<<<<<<<<<<<<<

#-------------------------------------------------------------------------------
# Usage: source HOSTS.CSH
#-------------------------------------------------------------------------------
# The following environment variables identify each AIPS host or TV server in
# the local network.
#
#    HOST    Name of the machine (uppercase).
#
#    SITE    The local site name (uppercase).
#
#    HOSTS   List of all hosts which run AIPS at this site.
#
#    ARCH    Differentiates executables and libraries for different machine
#            architectures.  Recognized values are "CVEX", and "SUN4".
#
# Original: MRC 90/Nov/16
# Modified: MRC 91/Jun/11
#-------------------------------------------------------------------------------

# Define host name logicals.
  set host = 'hostname'
  setenv HOST 'echo $host | tr a-z A-Z'

  set xxx = 'grep "^[+-]  $HOST" $AIPS_ROOT/HOSTS.CSH'

  if ("$xxx" == "") then
     setenv ARCH UNKNOWN
     setenv SITE UNKNOWN
     exit
  endif

  setenv ARCH $xxx[3]
  setenv SITE $xxx[4]

  set xxx = 'grep "^+.*$SITE" $AIPS_ROOT/HOSTS.CSH | awk '{ print $2 }''
  setenv HOSTS "$xxx"
  unset xxx

  exit 0

-------------------------------------------------------------------------------
Data section:  AIPS hosts are marked with "+", TV servers which don't run AIPS
               are marked with '-'.
```

```
+  ATELES       CVEX     EPPING        Convex C220
+  APUS         SUN4     EPPING        Sun 4/75 GX (room 94)
+  BOOTES       SUN4     EPPING        Sun 4/40 (Bob Sault)
+  BROGAR       SUN4     EPPING        Sun 4/40 (Ray Norris)
+  CARINA       SUN4     EPPING        Sun 4/60 GX (room 62A)
-  CRUX         SUN4     EPPING        Sun 4/65 (Computer room)
+  DYNAMO       SUN4     EPPING        Sun 4/20 (Tony Beasley)
+  FORNAX       SUN4     EPPING        Sun 4/40 (Eric Greisen)
+  GRUS         SUN4     EPPING        Sun 4/40 (Mark Calabretta)
-  HARMONIX     SUN4     EPPING        Sun 4/60 (John Bunton)
+  LUPUS        SUN4     EPPING        Sun 4/40 (Neil Killeen)
+  MAYHEM       SUN4     EPPING        Sun 4/20 (Henrietta May)
+  MENSA        SUN4     EPPING        Sun 4/60 (room 94)
+  OCTANS       SUN4     EPPING        Sun 4/60 (room 94)
+  PHOENIX      SUN4     EPPING        Sun 4/370 GX TAAC (room 62A)
-  POLYFONIX    SUN4     EPPING        Sun 4/60 (Ralph Marson)
+  PUPPIS       SUN4     EPPING        Sun 4/40 (room 62A)
+  VELA         SUN4     EPPING        Sun 4/60 (room 62A)
+  WYVERN       SUN4     EPPING        Sun 4/40 (John Deane)

+  CORVUS       SUN4     CULGOORA      Sun 4/65 GX
+  HYDRUS       SUN4     CULGOORA      Sun 4/75 GX
+  VOLANS       SUN4     CULGOORA      Sun 4/65

+  PICTOR       SUN4     PARKES        Sun 4/65 (control room)

>>>>>>>>>>>>>>>>>>>>>>>>>> $AIPS_ROOT/AIPSPATH.CSH <<<<<<<<<<<<<<<<<<<<<<<<<

#-------------------------------------------------------------------------------
# Usage: source AIPSPATH.CSH
#-------------------------------------------------------------------------------
# AIPSPATH defines the version of AIPS in use and redefines $PATH to include
# the AIPS system directories.  It also defines the AIPS programmer version
# selection macro environment variables.
#
# Original: MRC 91/Jun/08
# Modified: MRC 91/Jul/31
#-------------------------------------------------------------------------------
# AIPS home directory (hard-coded into START_AIPS and START_SERVERS also).
  setenv AIPS_ROOT /AIPS

# Get host info if necessary.
  if (! $?ARCH) source $AIPS_ROOT/HOSTS.CSH

# Define AIPS versions.
  setenv OLD $AIPS_ROOT/15APR91
  setenv NEW $AIPS_ROOT/15APR91
  setenv TST $AIPS_ROOT/15APR91

# Assign default version (usually set by START_AIPS).
  if (! $?VERSION) then
     setenv VERSION NEW
  endif

  switch ($VERSION)
     case OLD:
        setenv AIPS_VERSION $OLD
        breaksw
     case TST:
        setenv AIPS_VERSION $TST
        breaksw
     default:
        setenv AIPS_VERSION $NEW
        breaksw
  endsw

# LOCAL directory tree.
  setenv LOCAL ${AIPS_VERSION}_LOCAL

# Define AIPS procedure areas.
  setenv LOCSYSUNIX $LOCAL/SYSTEM/UNIX
  setenv SYSLOCAL $LOCSYSUNIX/$ARCH

# Save PATH as set on normal login and redefine it.
  if (! $?TPATH) then
```

```
    setenv TPATH "$PATH"
  endif
  setenv PATH .:"$SYSLOCAL":"$LOCSYSUNIX":"$TPATH"

# Define AIPS programmer version selection "macro" environment variables.
  setenv CDOLD "eval setenv VERSION OLD; \
    source ${OLD}_LOCAL/SYSTEM/UNIX/CDVER.CSH"
  setenv CDNEW "eval setenv VERSION NEW; \
    source ${NEW}_LOCAL/SYSTEM/UNIX/CDVER.CSH"
  setenv CDTST "eval setenv VERSION TST; \
    source ${TST}_LOCAL/SYSTEM/UNIX/CDVER.CSH"

>>>>>>>>>>>>>>>>>>>>>>>>>> $AIPS_ROOT/AIPSASSN.CSH <<<<<<<<<<<<<<<<<<<<<<<<<

#-------------------------------------------------------------------------------
# Usage: source AIPSASSN.CSH
#-------------------------------------------------------------------------------
# AIPSASSN defines environment variables required to run AIPS.  It relies on
# assignments made by AIPSPATH.CSH, particularly $AIPS_ROOT.
#
# Explanation of some of these variables:
#
#    NET0           Directory containing the network system parameter file
#                   (NETSP).  Used by ZDCHIN.
#
#    DA00           Host-specific directory containing AIPS system files.
#
#    HOSTID         A unique two-digit number for each AIPS host at each SITE
#                   used for scratch files.  The number is determined by the
#                   order of HOST in HOSTS.
#
#    LWPRINTER      Laserprinter to use (default is "laser", see ZLASCL and
#                   ZLPCL2 in $LOCSYSUNIX).
#
#    MT0m           Environment variable set by ZMOUN2 when the tape device
#                   corresponding to TAPEn (m = n - 1) is mounted.
#
#    TAPEn          Available tape devices (raw, no-rewind device file).
#
#    TAAC1          If set, this indicates that a TAAC graphics accelerator
#                   board is installed, and points to the home directory of
#                   the TAAC software.
#
#    RESSTTn        Terminals reserved for AIPSn (if any).  Used by ZSTRTA.
#
#    TASKTTn        AIPSn task message terminal (if any).  Usually associated
#                   with reserved terminals.  Used by ZTTOPN.
#
#    TEKTKn         AIPSn Tektronix 4010/4012 compatible terminal (if any).
#
#    BATCH_OUT      Batch AIPS error file/terminal.
#
#    LPRNTR         Line printer spool directory.  Used by ZLPOPN.
#
#    PLOTTER        Plotter spool directory.  Used by ZLASIO and ZLWIO.
#
# Original: MRC 91/Jun/08
# Modified: MRC 91/Jun/11
#-------------------------------------------------------------------------------
# Miscellaneous non-version specific directories
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Ionospheric data.
# setenv AIPSIONS $AIPS_ROOT/TEXT/IONS

# Public.
# setenv AIPSPUBL $AIPS_ROOT/TEXT/PUBL

# Gripes.
# setenv AIPSGRIP $AIPS_ROOT/TEXT/GRIP

# AIPS user lists.
# setenv AIPSWHO  $AIPS_ROOT/TEXT/WHO

# MSGS directory.
  setenv MSGS $AIPS_ROOT/SITES/$SITE/MSGS
```

```
# BOOK directory.
  setenv BOOK $AIPS_ROOT/BOOK

# Programmer home directories.
  if ("$SITE" == EPPING) then
     setenv EWG  /mnt/egreisen/AIPS
     setenv HM   /mnt/hmay/AIPS
     setenv MRC  /mnt/mcalabre/AIPS
     setenv NEBK /book/nkilleen/AIPS
  endif


# Non-version specific directories used by AIPS
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# System OFM files.
  setenv AIPSOFM $AIPS_ROOT/TEXT/OFM

# User OFM file area.
  setenv OFMFIL $AIPS_ROOT/OFM

# User RUN file area.
  setenv RUNFIL $AIPS_ROOT/RUN

# PRINT spooling area.
  setenv PRTFIL $AIPS_ROOT/PRINT

# PLOT spooling directory.
  setenv PLTFIL $AIPS_ROOT/PLOT


# System and data areas
# ~~~~~~~~~~~~~~~~~~~~~
# System areas.
  setenv NET0 $AIPS_ROOT/DA00
  setenv DA00 $AIPS_ROOT/DA00/$HOST

# Default data areas (may be reset by DADEVS.CSH).
  setenv DA01 /DATA/DA01
  setenv DA02 /DATA/DA02
  setenv DA03 /DATA/DA03
  setenv DA04 /DATA/DA04
  setenv DA05 /DATA/DA05

# FITS disk area.
  setenv FITS /DATA/FITS


# Version specific directories
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Memory areas.
  setenv OLDMEM  $OLD/MEMORY/$HOST
  setenv NEWMEM  $NEW/MEMORY/$HOST
  setenv TSTMEM  $TST/MEMORY/$HOST

# Load areas.
  setenv LOAD    $AIPS_VERSION/$ARCH/LOAD
  setenv REVERT  $LOCAL/$ARCH/LOAD
  setenv TVLOAD  $LOAD

  setenv OLDPSAP $OLD/$ARCH/LOAD/ALT1
  setenv NEWPSAP $NEW/$ARCH/LOAD/ALT1
  setenv TSTPSAP $TST/$ARCH/LOAD/ALT1

# Errors area.
  setenv ERRORS  $AIPS_VERSION/ERRORS/$HOST

# System run files.
  setenv RUNSYS  $AIPS_VERSION/RUN


# Host identification number
# ~~~~~~~~~~~~~~~~~~~~~~~~~~
  @ hostid=0
  foreach ahost ($HOSTS)
```

```
      @ hostid++
      if ("$ahost" == "$HOST") break
    end
    setenv HOSTID "$hostid"
    unset hostid ahost


# Device assignments
# ~~~~~~~~~~~~~~~~~~
# Tape drives initially unmounted.
    setenv MT00 UNMOUNTED
    setenv MT01 UNMOUNTED
    setenv MT02 UNMOUNTED
    setenv MT03 UNMOUNTED

# Determine controlling terminal for ZSTRTA.
    setenv TASKTT0 "'tty'"

# Printer and plotter definitions.
    setenv BATCH_OUT /dev/console
    setenv LPRNTR  /tmp
    setenv PLOTTER /tmp


# Host specific definitions
# ~~~~~~~~~~~~~~~~~~~~~~~~~
# LWPRINTER, TAPEn, TASKTTn, RESSTTn, TEKTKn, TAAC1,
# and re-definition of any of the foregoing.

    if ("$SITE" == EPPING) then
#     ATNF Epping.

      if (! $?LWPRINTER) setenv LWPRINTER "'lwdefault'"

      if ("$HOST" == ATELES) then
        setenv TAPE1 /dev/rmt00
        setenv TAPE2 /dev/rmt01
      else if ("$HOST" == APUS) then
        setenv TAPE1 /dev/nrst0
        setenv TAPE2 /dev/nrst1
      else if ("$HOST" == CARINA) then
        setenv TAPE1 /dev/nrst0
      else if ("$HOST" == PUPPIS) then
        setenv TAPE1 /dev/nrst0
      else if ("$HOST" == PHOENIX) then
        setenv TAAC1 /home/taac
      endif

    else if ("$SITE" == CULGOORA) then
      setenv LWPRINTER laser

      if ("$HOST" == CORVUS) then
        setenv TAPE1 /dev/nrst1
      else if ("$HOST" == HYDRUS) then
        setenv TAPE1 /dev/nrst1
      endif

    else if ("$SITE" == PARKES) then
      setenv LWPRINTER laser

      if ("$HOST" == PICTOR) then
        setenv TAPE1 /dev/nrst1
      endif

    endif

>>>>>>>>>>>>>>>>>>>>>>>>>>>>> $AIPS_ROOT/START_AIPS <<<<<<<<<<<<<<<<<<<<<<<<<<<<

#/bin/csh
#-------------------------------------------------------------------------------
# Usage: START_AIPS [OLD, NEW, or TST] [TV device] [DA_host:host:...]
#                   [options passed to AIPS]
#-------------------------------------------------------------------------------
# csh script to define AIPS system variables and initiate AIPS for a user from
# a general account.  The user must be in the aipsusr group in /etc/group.
```