AIPS INSTALLATION SUMMARY

For the 15OCT99 *Release*

Patrick P. Murphy
National Radio Astronomy Observatory
520 Edgemont Road
Charlottesville, VA 22903–2475, USA
pmurphy@nrao.edu

Requests for help, etc. to aipsmail@nrao.edu
or call (804) 296-0372, or fax (804) 296-0276
(email preferred)

World Wide Web accessible at
http://www.cv.nrao.edu/aips/

Ignore this page.

TABLE OF CONTENTS

Ignore this page.

1. INTRODUCTION

This document is intended as a guide for the person installing *AIPS* on a Unix† system. More specifically, it is aimed at those systems to which *AIPS* has already been ported, and for which sets of system-specific routines, shell scripts, and in some cases, precompiled binaries ("executables", compiled programs or tasks) are available. This currently includes the following (in alphabetical order):

- Alpha AXP systems (Linux 2.x)
- Alpha AXP systems (OSF/1 "Digital Unix" 4.0)
- Hewlett-Packard 9000/7xx series (HP-UX 10)
- *IBM RS/6000 AIX systems (AIX 3.2)*
- Intel 386/486/Pentium systems (Linux 2.x, ELF format, glibc)
- Silicon Graphics systems running IRIX (version 5 or 6)
- Sun Sparc systems (SunOS 5.x [Solaris 2]) including Ultras
- *Sun Sparc systems (SunOS 4.1 [Solaris 1])*

(Italics indicate older systems that we no longer support for our in-house users). To a lesser extent it may also cover Alliant, Cray, Convex, and Ultrix systems, though NRAO no longer has easy access to any of these systems. A port to FreeBSD or NetBSD would be trivial, but for these and any other systems not listed above, there are certain additional steps necessary (editing and creating scripts, creating system-specific Z routines, *etc.*) that are not covered here. There is a separate document, the *AIPS Unix Porting Reference*, that covers more of what is needed for a new port, but it has suffered from neglect for some years and is quite out of date.

For the systems listed above, the instructions here are for the most part all you need to perform an *AIPS* installation. However, occasional references are made to the *AIPS Unix Porting Reference* (though these have now been minimized). Often, referring to the source of the scripts or system routines can clarify points of confusion so it may not be necessary to consult this older document at all.

The *AIPS* installation consists of four major steps; these are summarized in section 4, and described in detail in the sections that follow. If you are installing a binary distribution (*e.g.*, from CD-ROM, from a binary tape or by downloading binaries automatically via the internet), there is only one major step. This document attempts to describe what is necessary for both cases. Some details of each major step are listed following the summary. The first step is the most complex while the remainder are generally automatic — and they are for the most part checkpointed, so a failure in the middle of any step does not usually mean restarting the entire step.

Unix Environment variables are indicated in this document as you would type them interactively, *i.e.* with a dollar sign preceding the variable name. An example would be "`echo $AIPS_ROOT`" to show the value of the `$AIPS_ROOT` environment variable.

---

† While there is some archæological support in the distribution for non-Unix systems, specifically VMS, it is almost certainly non-functional.

2. NEW SYSTEM FEATURES FOR 15OCT99

If you have previously installed any version of *AIPS* since (and including) the 15APR92 version, this section is for you. If you are new to AIPS, you may want to first read the section below on the Network Setup, then return and read this section.

- SOL86 is now supported as a regular architecture. This is for Intel systems running Solaris. The *AIPSLetter* has the details including web URLs for more information, but suffice it to say here that the settings for this architecture as shipped assume the GNU compilers.
- This is the last planned public release of *AIPS*. However, the "next" version (what we call TST here within NRAO) has been labelled 31DEC99 and nicknamed "*AIPSfor the Ages*" (or Aged, depending on your perspective!). This version is now our "midnight job" version, it continues to change on a daily basis, and it is served via a secure shell (ssh) based system to sites both within NRAO and outside users who ask for it. The main *AIPS* web page has the details on how to get 31DEC99 and install or upgrade to it.

The "midnight" job refers to a set of scripts that allows any *AIPS* installation to keep current with the master version here in Charlottesville. It can be run daily, weekly, or on demand depending on the needs of your user(s). Setup of this job has been considerably simplified, and scripts are (or will shortly be) available to minimize the effort needed for installation.

For additional information, read through the document. For application-level changes, refer to the *AIPSLetter* for the release. Also check for patches and additional last-minute information on the *AIPS* web pages, which are accessible on the world wide web at http://www.cv.nrao.edu/aips/.

2.1 Architecture-specific Notes

The purpose of this section is to point out various quirks of the different architectures that were encountered during the installation testing, and that various people have reported back to the *AIPS* group (thank you!). It is probably inadequate, but better than nothing. Most of what is presented in this section will only make sense after you have read the rest of the document, or if you are a seasoned *AIPS* installer. The architectures are presented in alphabetical order.

2.1.1 Alpha (Digital Unix)

The DEC Fortran compiler emits some informational messages when compiling the PP.FOR and UNSHR.FOR utility programs. Also, when building the readline library, some "*Invalid configuration; machine not recognized*" errors may appear. All these turn out to be harmless as the library and programs will build (under OSF1 4.0D at least) without serious errors.

2.1.2 Alpha (Linux)

Recently, Compaq have released both a version of their math library for Alpha Linux, and a beta version of their traditional (DEC) compilers. We strongly suspect that use of these compilers or this library will significantly enhance the performance of *AIPS* on the Alpha/Linux platform, but as of yet we have not had a chance to put this to the test. Changes to LIBR.DAT and the various options files (CCOPTS.SH and LDOPTS.SH in $SYSLOCAL, FDEFAULT.SH in $SYSUNIX) would be necessary in order to try these; if any of you do try, please let us know how it went.

2.1.3 Hewlett-Packard 9000/7xx

If you run the amd automounter, INSTEP1 may have problems with the $AIPS_ROOT section as it is likely to detect a dot in the pathname. In this case, it may be necessary to hand-edit INSTEP1 to hard-wire the definition of your $AIPS_ROOT area.

2.1.4 IBM RS/6000

NOTE: The AIPS group no longer has an RS/6000 system running AIX. No binaries were generated this year, and no install testing was performed for the IBM architecture. We have NOT tested *AIPS* under AIX 4. The rest of this section dates from the 15OCT98 release, the last one where we actively supported this architecture.

It may be necessary to increase the limit on maximum number of simultaneous processes per user under AIX if your users do a lot of printing. This is caused by the deferring of the temporary file deletion within *AIPS* (done via a fork). Your system administrator can use smit to change this.

Also, be careful with using the PSALLOC variable. We have found it next to impossible to do anything meaningfull with this set so that it forces conventional behaviour on the malloc() system call. If you run programs *without* PSALLOC set to early, and your system gets tight on resources, the kernel starts killing off random processes. On the other hand, having it set means it takes enormous memory and swap resources to do the simplest things.

Make sure you use the cc compiler and not the xlc version.

If the -Pv compiler directive is used (as it is automatically for AIX version 3.2), this causes a special IBM preprocessor to be used. This in turn produces non-ANSI Fortran, which generates warnings because the -qfips qualifier forces ANSI compliance. Presumably the benefits of the resulting optimization are worth the irritation of the warnings.

2.1.5 Intel Linux

The install testing for 15OCT99 was carried out on a Linux 2.2.5 system basedon Red Hat 6.0. It used the egcs version 1.1.2 compilers and glibc (the GNU version of the C library).

If you have a distrubition other than Red Hat (e.g. Debian), you may need to edit $SYSLOCAL/LIBR.DAT and replace all occurences of -ltermcap with -lncurses. This is needed to recompile (INSTEP2, INSTEP4) the source. The binaries built on our Red Hat systems apparently do not work without the libtermcap.so shared library.

The system calls needed by the GNU Readline library are supplied by the `ncurses` library on Debian, unlike RedHat and others where the routines are in the `termcap` library.

For source-only installations on Red Hat (*i.e.*, not CD-ROM or binary), you need to ensure that `libtermcap.a` is installed on your system. On stock RedHat 5 or 6 systems it may not installed by default. Type `rpm -q libtermcap` to see if you have it. If not, use the Red Hat CDROM or various ftp sites to install it, or have your sysadmin do it for you. You need to install this library before running `INSTEP3` or `INSTEP4`.

For Debian/GNU distributions, you may need the `ncursed-dev` package. If you do, this need will become apparent when/if you try to build `readline`. Specifically, the `termcap.h` file needs to be present.

In the somewhat unlikely event that you are building on a 386 system, you should remove the "`-O2 -m486`" from the relevant options line in the `XAS` Makefile prior to building it (or having `INSETP1` build it for you). Apparently this option improves performance somewhat on 486 or better processors, but does not work on Intel 80386 based systems.

If you are running a version of Linux prior to 1.3 (*i.e.*, using `a.out` object file format and not ELF format), and have a version of `gcc` or `ld` of 2.6.3 and 2.5.2.6 (with BFD 2.5) respectively or later, then *please update your system*! You will have many, many problems with these older versions of the kernel and libraries, and several tasks will not build correctly.

If you have a standalone PC that is not connected to any network, either via ethernet or phone (or you want to run without tying up a phone link), you still may need to set up a dummy "loopback" or `lo` network interface so that the internet services `SSSIN` and others will work. Use of `aips tv=local tpok` to start *AIPS* will get around this as it causes the display servers to use Unix sockets, and suppresses starting of the `TPMON` dæmons. See the section below (7.5) on setting up services.

2.1.6 Silicon Graphics

With the version of `readline` included in this *AIPS* distribution, it may be necessary to set the `STRIP` option to `FALSE` in the `$SYSLOCAL/LDOPTS.SH` file. This was true on the test-install at NRAO where Irix 6.4 was used on an Origin 200.

It is necessary to have `/usr/bsd` in your `PATH` environment variable prior to starting the installation.

If you use `gcc` as a compiler for creating the Readline library (its configure script tries hard to find it), you will probably have to force the make to use `CFLAGS="-o"` as some versions of GNU C do not fare well with the default flag settings of `-g -o`. The making of the readline library will still produce many warnings; however, they are just that: warnings. The library builds successfully, and the readline code in `AIPS` works as it should.

In addition, if you do not have a multi-processor system or the multiprocessor library support, you will have to comment out the `-mp` flag in `$SYSUNIX/FDEFAULT.SH` in the `SGI` section.

If you do not have GNU make, you will have to edit the `Makefile` for `XAS` by hand for use with the stock `make` supplied with the operating system. Simply replace all occurences of the string $(ARCH) with SGI, then type make xas followed by make install and make clean. It still produces many, many warning messages (with or without GNU make), but produces a working program.

Several times during the test install, a directory name may be unexpectedly printed in the middle of the usual INSTEP1 verbosity. This may have been an artifact of the local account or SGI system, or it may be endemic to IRIX. In either event, it apparently produced no nsignificant problems.

2.1.7 SunOS Version 5

NOTE: The AIPS group no longer has any Sparc systems running SunOS 4. No binaries were generated this year, and no install testing was performed for the SUN4 architecture.

This section covers both SOL and SUL architectures; the latter only differs from the former by the Ultra-specific optimizations performed in the compile stage.

If an *AIPS* data area is accessed through the SunOS 5.3 automounter, access to the area the first time after it has been unmounted (automatically) may fail with "file not found". The quick fix is to try again as the first attempt will have mounted the area. The more permanent fix is to upgrade. Newer versions of SunOS do not have this problem (and they run faster too, so upgrade!).

If you obtained the binary distribution and do *not* have a Fortran compiler, be sure to place the libF77.so.3 and libsunmath.so.1 shared libraries (which you can find either in your $AIPS_ROOT/SOL/lib directory for tape binary installations, in $AIPS_ROOT for ftp binary installs) in /opt/SUNWspro/lib. This is necessary so that the shared library can be detected in the correct place by the dynamic loader ld.so at runtime. If you cannot place it here, try putting it in $SYSLOCAL or $AIPS_ROOT, and include this area in the definition of $LD_LIBRARY_PATH in your .login or .profile initialization file. It is in general better to not have this environment variable defined at all, as it causes many problems.

In fact, before starting INSTEP1 it is strongly recommended that you unset this variable (use unset $LD_LIBRARY_PATH for bourne-like shells (bash, zsh, ksh, sh) and for c-like shells, unsetenv $LD_LIBRARY_PATH). If this is not feasible for you, then at least make sure you do *not* have /usr/ucblib in the definition of this variable before starting the install (or if you must have it, be sure to put /usr/ccs/lib before it). Failure to heed this warning will result in multiple AIPS1's and general confusion, not to mention a real mess with file locking.

Make sure the environment variable OPENWINHOME is defined correctly before starting INSTEP1. Also, ensure that $OPENWINHOME/bin is in the path. Otherwise you will have problems building XAS. If you don't use OpenWindows for X11, please define the variable anyway so that it points at your main X11 area, *e.g.*, /usr/local/X11.

The install test for SunOS 5 was performed with the SC4 compilers from Sun (4.2 to be precise). Other vendors' compilers have been reported to cause significant problems (including catastrophic failure of the DDT test). If you use any compilers other than those from Sun, you need to change the compiler qualifiers used in the $SYSUNIX/FDEFAULT.SH, $SYSLOCAL/CCOPTS.SH, and $SYSLOCAL/LDOPTS.SH files prior to INSTEP2, and in fact you should consider it a new port, not just a minor variation on the existing Solaris port. Should you get such a non-Sun compiler to work, please contact the AIPS group with your success story! This does not refer to g77; the *AIPS* group has found this to work, albeit producing binaries that ran slower than those using the Sun compilers.

To avail of the larger (1280×1024) screen sizes that some Sparc Ultra systems now come with, you may want to increase the default shared memory segment size from its default of 1 megabyte. Adding this line in your /etc/system file should work:

```
set shmsys:shminfo_shmmax=10485760
```

This increases it to 10 megabytes; less may work fine for you. The XAS image display server will request shared memory segment sizes that are related to the frame buffer size.

Finally, make sure you read the section entitled "*Improving AIPS Performance Under Solaris*" in the 15APR99 *AIPSLetter*; it has advice on how to enhance the effective *AIPS* performance on suitably equipped systems (*i.e.*, lots of memory). In a nutshell, adding something like this:

```
set ufs:ufs_HW=6291456
set ufs:ufs_LW=4194304
```

to your /etc/system file may improve performance anywhere from 10% to 30% as measured by the DDT benchmark. This is essentially a safer way of implementing the "tmp" trick referred to on the *AIPS* benchmarking web pages. Do *not* try these settings if you have 64 megabytes or less of memory.

3. NOTES ON THE MULTI-HOST (NETWORK) SETUP

If you are already familiar with the network setup, and/or if you have never seen a version of *AIPS* prior to 15APR92, you can probably skip this section. If your last exposure to *AIPS* was with the 15APR91 version or earlier versions, or if you are confused about how to share your *AIPS* installation among two or more systems, this section is required reading.

Starting with the 15APR92 release of *AIPS*, many of the concepts and techniques used by the ATNF (Australia Telsecope National Facility) in their *AIPS* setup were adopted in the master version in Charlottesville. For details on exactly what these are, see *AIPS* memo 74.

The most basic change is that the directories in *AIPS* that contained binary or configuration files specific to one architecture or host have been moved around. There are a set of directories under the 15OCT99/ area with names like LINUX and SOL that contain the architecture-specific things like the LOAD/ directory for the executables (binaries) and the LIBR/ area for object libraries. The $SYSLOCAL area has also moved under this architecture-specific area, as SYSTEM/SITENAME where you substitute your own site name.

The DA00 area for system files is now under a new area called $NET0 which expands to $AIPS_ROOT/DA00/, and the actual $DA00 environment variable expands to $NET0/$HOST or $AIPS_ROOT/DA00/$HOST (here $HOST represents your host name in uppercase). The *AIPS* startup scripts will *try to* figure out which TV to use; each TV has a name, usually the host name of the workstation that XAS actually runs on. You can of course override the default TV with a command line option (see the AIPS man page in $SYSUNIX/AIPS.L, or type HELP AIPS from inside *AIPS* itself). By using "local" TV's (which in turn use Unix sockets in place of INET sockets for communication between the TV and *AIPS*), you can run multiple instances of the TV on a single host. If you have a central compute server and many X terminals, this will be quite useful.

It is even possible to mix big- and little-endian systems in the same *AIPS* network (*i.e.*, sharing the same $AIPS_ROOT area). This is done, for example, at the Charlottesville NRAO site, where a growing network of Dec Alpha (OSF/1 and Linux) systems and many Intel PC (Linux) systems now manage to co-exist nicely with a much larger network of Suns, IBM RS/6000's and a couple of HP's. This is made possible via the $SITE environment variable; the Dec and PC systems are all little-endian and use IEEE floating point format; their entries in the HOSTS.LIST file are under site VCOARN (NRAOCV spelt backwards) whereas all the Suns and IBMS in the same file are listed as site NRAOCV. *However, big and little endian systems CANNOT share data areas* though they can share FITS disk files.

If you are reading this section, and have not read the previous section (section 2) please go back and do so now.

4. OUTLINE OF THE FOUR INSTALLATION STEPS

The installation process is divided into four separate steps, usually referred to as "in-steps" (from the name of the commands used to do them). The last three of these are only required for source-only tapes (or tar files) as they involve compiling and linking the programs from scratch. These last three are mostly automated, but the first "in-step" will require some work, most of it in the form of decisions on how to set up your *AIPS* environment, and editing configuration files.

4.1 INSTEP1: Unloading the Tape, Preliminaries

This step requires most of the *AIPS* installer's time. It involves unpacking the *tar* file from tape or disk, and then running a shell script called `INSTEP1`. For CD-ROM installations, the `CDSETUP` script is run instead; this in turn will call `INSTEP1` after setting up a skeleton directory structure (and symbolic links).

During `INSTEP1`, you will be asked a lot of questions, and the script will make a reasonable attempt to set up directories and configuration files. It may even offer to download the binaries for your system from NRAO, if we have them. `INSTEP1` works best on systems already supported in-house at NRAO (see the introduction for a list), but even if your system is not one of these, the procedure will still run.

For *CD-ROM*, *binary tape*, or *binary ftp* installations, you should be able to actually run *AIPS* after `INSTEP1` completes and you define the *AIPS* environment (by calling the relevant shell script `LOGIN.SH` or `LOGIN.CSH`). However, the TV (`XAS`) and other servers will not work if you have not defined the internet services for *AIPS*; see section 7.5 below on the `/etc/services` file. If you forget the services, you will get many, many confusing error messages on a normal `AIPS` startup, and of course no TV, tekserver or message server window. (You do not need most entries in `/etc/services` if using `tv=local` Unix based servers).

Finally, check the patch area on our web server (see section 5.5, *Get any Patches* below for details) and retrieve any necessary files before starting the next step. Note that patching binary installations may be difficult or impossible if you do not have the same compilers which were used by NRAO to build the binaries.

4.2 INSTEP2: Subroutine Compilations

This step is for source-only installations. It compiles all the *AIPS* subroutines and places the object files in about twelve separate libraries. The `INSTEP2` shell script will generate a set of list files with names like `AIPSUB.LIS`, `APLSUB.LIS`, *etc.*. These are processed by the `COMRPL` shell script, which provides the checkpointing mechanism. If there is a minus sign in front of any filename in a list file, that means it has been successfully processed.

If this step fails for any reason, inspect the tail end of the file `INSTEP2.LOG` and try to see which step failed (by searching backwards) and if possible, why. If there is no apparent reason, just try to restart it.

4.3 INSTEP3: Subset Program Compilation

This step is also for source-only installations. It compiles the AIPS main program and about 20 of the most commonly used tasks. The programs are linked with the *AIPS* libraries. The intent here is to enable the running of the *AIPS* benchmark and test suite known as the "DDT"*; this will give you a quantitative measure of the accuracy and speed of AIPS on your system. The DDT is a set of tasks with known results; the tasks are run and compared with reference data.

Before starting AIPS or running the DDT, you must run three programs: FILAIP, POPSGN and SETPAR to create files needed by the AIPS program. To avoid some error messages on your first AIPS startup, you will also have to make sure the network services are defined in /etc/services or your YP/NIS services map (see section 7.5).

For standard configurations mentioned in the introduction, running the DDT should not be necessary; you will know if your *AIPS* installation works by just starting up the system, running a task like DISKU and entering something like PRINT 2 + 2 (hint: the answer should be 4; this command will test quite a bit of the internal workings of AIPS, and should not be laughed at — at least not too loudly).

Running the DDT requires either a DDT tape, the CD-ROM or that you get the relevant DDT FITS files. These are available via anonymous ftp or the world wide web, or if necessary you can request a tape from NRAO.

Unlike the previous step, INSTEP3 calls the COMLNK procedure which links the object file against the libraries and produces a binary or "executable" file . This file is then moved to the $LOAD area. If you have configured your system for more than one type of TV device (very unlikely) or more than one AP-type device (*exceedingly* unlikely), there will be alternate load areas, and some tasks will be linked more than once.

If something goes wrong during INSTEP3, look at the log file INSTEP3.LOG and see if you can understand the reason for the failure. These log files tend to be quite verbose, so as long as you understand the underlying basic operations, it should be possible to trace down most problems. Watch out for disk space; a disk filling up is the most common source of problems here and with INSTEP4. Such disk problems can also occur if your /tmp area is too small or too full.

4.4 INSTEP4: All Other Programs

This step compiles and links all of the *AIPS* tasks. It will rebuild the tasks already done in INSTEP3, but this is only 20 out of 401 tasks. You should verify that it works (after having the internet services defined, see section 7.5) by starting AIPS and trying a few things like free, tvinit, go mandl, tvall, and so on. If you are uncertain that the system built correctly, the DDT test suite of programs can be run now.

If you have a version of *AIPS* older than 15JAN95, you will have to run the UPDAT program to convert your user data so that it can be read by 15OCT99. Once you convert a given user's data on a given disk, that data can no longer be accessed from the older version(s) (other than via FITS disk/tape).

* Dirty Dozen Tests; this is how many tasks were in the test originally. See AIPS memos 85 and 73 for details.

5. INSTEP1

In what follows, things you should type are presented between the typical Unix bourne- or korn- or bash-shell prompt `$` and the symbol ↓ for carriage return (or "Enter" on some keyboards). As an example,

```
$ you type this↓
```

means that all you type is `you type this` followed by pressing the `<return>` or `<enter>` key. Also, examples of interaction may be shown thus:

`INSTEP1: ===> Computer says this:` *and you type this* ↓

All queries that `INSTEP1` asks will be preceded by the `"===>"` string.

5.1 Unload Tape, Unpack File, or Insert CD

AIPS is provided for Unix systems in four forms: CD-ROM, source, binaries via ftp, and binary tape. *AIPS* is covered by the GNU General Public License; see section 11 below for the terms.

5.1.1 AIPS on CD-ROM

Since the `15APR98` release, NRAO offers a CD-ROM based distribution directly. A full source tree is present on the disk, and binaries for the two most popular architectures are also included — `SOL` and `LINUX`. The binaries are uncompressed and can be run "live" off the CD-ROM, or installed on your local disk. You can even switch back from one to the other in a convenient manner. If running off the CD-ROM, the minimum footprint on your disk will be about 10 Megabytes. This does not count user data areas!

The only difference between the NRAO CD-ROM distribution and the alternate methods (ftp, tape) is that you run a different script to get things started. This script, `CDSETUP`, will configure the initial set of files and symbolic links and then automatically start `INSTEP1`.

The `CDSETUP` script is located in the top level directory of the CD-ROM; in order to run it and the other shell scripts on the CD, you will need to enable execution of scripts and binaries. On some Linux systems you may need to override the defaults for the CD-ROM thus:

```
# mount -r -o exec /mnt/cdrom
```

For this to work, you may either have to be root, have your systems administrator do this for you, or have her/him add you to the "sudo" facility.

To receive an AIPS CD-ROM, point your web browser at

```
http://www.cv.nrao.edu/aips/forms/aipsorder.html
```

In addition to NRAO's CD-ROM distribution, a group of Linux enthusiasts ("*The Random Factory*") have periodically released a CD-ROM entitled "Linux for Astronomy". The latest version contains volumes 3 and 4, and the `15APR98` version of AIPS is contained thereon. A newer version, likely to include 15OCT99, is expected soon. It also has ASSIST, CLOUDY, GIPSY, *IRAF*, Karma, Midas, Nemo, PGPerl, PGPLOT, SAOtng, Starbase, Xephem, Xite and more (including a complete 2.0.30 based Linux distribution). If you are using that CD, refer to the instructions that came with it to get AIPS into a state where this document is useful (the binaries on CD are compressed). For additional information, contact the Random Factory at `rfactory@earthlink.net`, refer to their web page at `http://www.randomfactory.com/lfa.html` or call (520) 820-5221. NRAO has no formal ties or involvement with The Random Factory (though we cooperate with them and wish them well), and this notice is inserted here for the benefit of the Astronomical Community.

5.1.2 Source via FTP

The second method for getting *AIPS* is in source form, as a compressed tar file made available via *anonymous* `ftp` on the Internet. The `15OCT99` tar file is over 138 MBytes in size. Both a GNU-zipped (`15OCT99.tar.gz`) and compressed (`15OCT99.tar.Z`) file are provided; the former is 28.7 MBytes and the latter is 40.4 MBytes. The MD5 checksums on these files are as follows:

```
   15OCT99.tar : 1f0d039823f164066b2066bd6865e49d
15OCT99.tar.gz : 279c5f4625c359afb74b59476407f224
 15OCT99.tar.Z : 7730c6bc150c8277927f5517c2d6e715
```

The master version is on `aips.nrao.edu`, (currently 192.33.115.108, though this IP numeric address may change [it did in April 1998]; the name will not) in directory `/aips/15OCT99`.

The *AIPS* source is also available on magnetic tape in a variety of forms; see the "*Binary Tape*" section for details. Note that the other distributions of *AIPS* (CD-ROM, binary tape, binaries via FTP) also include the source code.

When you start the installation of AIPS from one of these "tarballs", you will be asked if you want to attempt to retrieve the binaries via anonymous FTP; doing so changes the installation from source-only to binaries via anonymous ftp; see the next section for more details. This can save you the time needed to compile and build the almost 400 individual tasks and 13 libraries.

5.1.3 Binaries via Anonymous FTP

The third form of distributrion is to obtain the source code and optionally the binaries via anonymous ftp. This option is still the most popular, with the CD-ROM distribution a close second. Getting binaries in this manner is only recommended for sites with good internet connectivity. `INSTEP1` automatically asks if you want to get the binaries from `aips.nrao.edu` via anonymous ftp. Even after compressing the files in the `LOAD` area with `gzip` as has been done since the `15OCT96` release, they can still amount to over 100 megabytes. So please think whether it is possible, realistic, and a good idea before opting for this method. If you choose the time to match off-peak hours at both your site and ours, you are more likely to get decent bandwidth and success in copying all files.

For the `15OCT99` release, binaries are provided for:

- `ALPHA`: Alpha AXP systems, OSF1 V4.0B

- AXLINUX: Alpha AXP systems, Linux 2.0.34 (ELF/glibc/egcs-1.1)
- HP: HP 9000/8xx HP-UX 10.01
- LINUX: Intel 386/486/Pentium systems, Linux 2.2.5 (ELF/glibc/egcs-1.1.2)
- SGI: Silicon Graphics MIPS-based systems, IRIX 6.4
- SOL: Sparc systems, SunOS 5.6 (SC4.2)
- SUL: Sparc systems, SunOS 5.6 (SC4.2), Ultra 2 optimized

Check the AIPS web pages for additional information; some of these may not be available. This operating system and compiler information reflects the system on which the binaries on the tapes were built. In many cases, if you have a more recent version of the operating system, *e.g.*, Linux kernel 2.2, the binaries (executables, programs) on the tapes will still work. For SunOS 5, if you do not have the same compiler as we do (see above), you *may* need to copy the Fortran run-time shared object libraries (`libF77.so.*` *etc.*) from where the installation leaves them in `$AIPS_ROOT`, to an appropriate area: preferably where we had it when the binaries were generated – `/opt/SUNWspro/lib` for SunOS 5 – or if you have no choice, some place in your default `$LD_LIBRARY_PATH` (note that for SunOS 5, use of this environment variable is *strongly discouraged*).

In general, these run-time libraries are *only necessary* if you do not have Fortran installed on your system, if you have a different version than indicated above, or put the compilers in a location other than the locations mentioned. It may need to be done on each host if you have many in your AIPS site, depending on where you place the libraries.

5.1.4 Binary Tape

The final form in which *AIPS* is provided is the *load-and-go* or `binary` tape. Formats on which it is available include: 4mm DDS DAT, or 8mm Exabyte (2G or 5G formats). We have discontinued the Quarter inch Cartridge and 9-track tape (6250 BPI) distributions due to low (or nonexistent) demand. The architectures available are the same as for Binaries via Anonymous FTP.

One difference between the binary tape distribution and getting the binaries via ftp is the location of the shared Fortran run-time libraries; if you unload from tape you will find them in `$AIPS_ROOT/BIN/SOL/lib` for Solaris.

5.1.5 Getting Started

The source code is provided on both source-only and binary distribution (including CD-ROM), and is contained in many sub-directories. Move to the directory where you intend on putting the `15OCT99` version of *AIPS*. This directory will henceforth be referred to as the `$AIPS_ROOT` area and it is convenient to have it be the login area of your `aips` account if you have one (or if you plan on having one). In the remainder of this document, replace `/home/aips` with the name of your directory. If you intend to install *AIPS* on more than one computer or workstation, remember to choose a directory for `$AIPS_ROOT` that will be visible via NFS* from all machines. For example,

* Sun's Network File System, used to mount a disk from one machine so that it is transparently visible from another.

at NRAO/Charlottesville, on all machines the directory is /AIPS and is NFS-mounted on aips hosts as /AIPS from a file server. It is important that the directory tree appears the same on each host in your network (*i.e.*, use the same mount point for $AIPS_ROOT).

If you are installing from tape, insert the tape in the drive and type:

```
$ cd /home/aips↓
$ tar xvf /dev/rmt/0↓
```

However, before doing this, note the extra mv commands recommended if you are installing over an existing *AIPS* installation (in section 5.2, about two pages beyond this point).

The above example is what one would typically use on a Sun running SunOS 5, with one tape drive, and assumes that /home/aips is your $AIPS_ROOT directory; substitute your actual directory name and the correct tape drive name as appropriate. Also, the second command (moving the old TEXT and BIN areas out of the way) is only necessary if you are installing a newer version "on top" of an older one.

For other systems, tape drive device names may be as follows:

⋆ Sparc (SunOS 4), /dev/nrst0 for first SCSI tape, /dev/nrst1 for second, *etc.*.
⋆ HP and SGI (Irix 5) and other System V Unices: /dev/rmt/0ln or /dev/rmt/1ln, *etc.*.
⋆ IBM RS/6000: /dev/rmt0 or /dev/rmt1, *etc.*.
⋆ DEC (Ultrix or OSF/1, Digital Unix): /dev/nrmt0h or /dev/nrmt1h, *etc.*.
⋆ Linux: /dev/nst0 generally.
⋆ Convex: /dev/rmt12 or similar; you also need to use tpmount.

These are just guidelines and your system may be different. Check with your system administrator or manager for details on tape device names if you are not sure.

If the tape device is on a remote system you can still read the files from it with a command like:

```
$ rsh <mach> 'dd if=/dev/rmt/0ln bs=20b' | tar xvBfb - 20↓
```

(if using the secure shell, replace rsh with ssh, and on HP systems, replace rsh with remsh). This command simply pipes what is on the tape across the network. You should replace the string <mach> with the name of the computer that the tape is on, and use the right /dev/xxx device name for the tape. For rsh to work, you may need to have your .rhosts file configured correctly; check your local manual pages.

If you are obtaining *AIPS* via Internet ftp, make sure you retrieve and *read* the README files in the /aips and also the /aips/15OCT99 directories before proceeding. That is where last-minute "READ ME FIRST" instructions will be posted. Then, unless such instructions indicate otherwise, switch your ftp program to binary mode, and retrieve either 15OCT99.tar.Z or 15OCT99.tar.gz. The latter is significantly smaller; if you are using one of the systems for which binaries are available, please download the gunzip binary from the relevant LOAD area, *e.g.*, 15OCT99/SOL/LOAD/gunzip and getting the .tar.gz file.

If you have problems transferring the single monolithic file, try getting it in parts. The README files indicate where you can get these files split into 50 equal parts; the `prompt` command should then be given to ftp so as to disable interactive prompting, and the `mget` command used to retrieve the 50 different parts. Then you may "cat" them together into one large file.

Once you have the file in one piece, but before unpacking the resulting `tar` file, remember to move any existing `TEXT` and `BIN` areas out of the way; also delete any existing `INSTEP1` and `REGISTER` symbolic links. Then you can unpack it via either:

```
$ gunzip -c 15OCT99.tar.gz | tar xvf -↓
```

or, if you have GNU tar (*e.g.*, for Linux):

```
$ tar xvzf 15OCT99.tar.gz↓
```

For the compressed version:

```
$ zcat 15OCT99.tar.Z | tar xvf -↓
```

Refer to your system documentation (manual pages or printed manuals) for more details on the `tar` and `zcat` commands. The `gzip`/`gunzip` utility is part of the GNU utilities, available free of charge on internet (*e.g.*, at `prep.ai.mit.edu`); ask your local network manager or expert for details.

If you obtained a binary tape distribution, you now need to uncompress the binary files. This can be done simply, for example:

```
$ cd 15OCT99/$ARCH/LOAD↓
$ ./gunzip *.gz↓
```

where you substitute `SOL`, `SUL`, `ALPHA`, `LINUX`, `AXLINUX`, `SGI`, or `HP` for `$ARCH` in the example above. If you have your own copy of `gunzip`, you can of course use it instead of the version supplied on the tape.

These commands will create four files and two symbolic links (six files if you have a tape) in `$AIPS_ROOT`: the `15OCT99` directory which holds almost all of *AIPS*; the `TEXT` area for ionospheric data, *AIPS* documents and OFM (color lookup) tables; a `BIN` area (used for some utility programs on the CD-ROM and "binary" tape); an empty `DA00` directory, and symbolic links to the `INSTEP1` and `REGISTER` procedures.

For source-only installations, the tar file for `15OCT99` will uncompress to just over 138 megabytes, including the `TEXT` and `15OCT99` directories. The CD-ROM, binary tapes and ftp distributions will have extra files which need considerably more disk space under the `15OCT99/$ARCH` areas (for binaries, libraries *etc.*). Generally this will vary from just under 200 Megabytes on Linux systems to over 300 Megabytes on SGI systems, once the binaries have been unzipped.

Add about 10 Megabytes to the above figures for system files, and 138 Megabytes for the contents of the tar file itself (double this if the tar file is on the same disk, not a separate one or tape) to get the **bare minimum** of disk space you will need to install AIPS. User Data Storage is **not** included in these estimates, and if you are building *AIPS* from source, you will need more space for temporary storage during compilation.

NOTE: If installing from a CD-ROM and you choose to run from the CD, you only need about 10 megabytes of local space.

Finally, a word of warning. During the install testing, one often has to be creative in allocating disk space. As an example, suppose the `15OCT99/SOL` directory was in fact a symbolic link (symlink) to a directory on a different filesystem. This can have the undesired side effect of making the reference to UNSHR in the building of XAS fail because a relative path was used. It is impossible to anticipate all possible problems that might arise due to the use of such symlinks; watching for problems and being aware of the symlinks is probably the best approach. The best approach, especially in this era of mind-bogglingly large and inexpensive disks, is to not use symlinks in this manner at all.

5.2 Installing over older AIPS installations

Here is a quick recipe for what you need to do prior to unpacking the 15OCT99 tape or tar file. Note that you should do the installation in the same $AIPS_ROOT as you used before, if at all possible.

```
$ cd $AIPS_ROOT↓
$ mv TEXT TEXT.OLD↓
$ rm REGISTER INSTEP1↓
$ mv BIN/$ARCH BIN/$ARCH.OLD↓
```

Once your installation is complete, you can delete the various .OLD directories to save space. Some of these areas, specifically BIN/$ARCH, may be empty or missing, but if not they can cause complications for INSTEP1.

5.3 Start the ball rolling: the INSTEP1 procedure

This procedure will do most, but not all, of the setup required for a given system or network configuration. In case there are problems with it, you may want to refer to the now considerably out-of-date *AIPS Unix Porting Reference* for additional instructions, or (better still) look at the source for INSTEP1 itself.

Before starting, you should make sure that your PATH environment variable includes certain areas. Under SunOS 4 and 5, in particular, you should have the OpenWindows directory included (usually /usr/openwin/bin; alternately you may use your local copy of X11), and also the area where the compilers are found (either /usr/lang, or for Solaris /opt/SUNWspro/bin). Another area needed for Solaris and OSF/1 systems is /usr/ccs/bin. On SGI systems, you need /usr/bsd in the path. If these are not in your PATH variable, you should modify your login file(s) (.login for csh and tcsh; .profile for bash, ksh, sh and zsh) so that the areas are automatically included for all future logins, and either log out and in again to make it take effect or (if you know how) redefine your PATH interactively before starting INSTEP1.

There are other environment variables you may want to define. On any system where Sun's OpenWindows is used as the source for the X11 libraries and include files, make sure to have the environment variable OPENWINHOME defined. This is normally set to /usr/openwin, though if you prefer the MIT X11 system (and you have it), you can define this to point at it instead. INSTEP1 has the ability to call up your text editor to edit various files during the procedure. If you have the environment variable EDITOR set to some value (*e.g.*, emacs or vi), the INSTEP1 script will use it in appropriate places. If it is not defined, you may want to define it prior to starting. For emacs, watch whether you need to define the DISPLAY environment variable.

When you are ready, make sure you are in the $AIPS_ROOT area (*e.g.*, by typing cd /home/aips). Then, if you are installing from a CD, mount the CD-ROM (usually on /mnt/cdrom on Linux or /cdrom/cdrom0/ on Solaris), and then type, for example:

```
$ /mnt/cdrom/CDSETUP↓
```

This will run a script that sets things up for INSTEP1 and then hands control over to it.

If you are installing from tape or FTP, you instead type this:

```
$ ./INSTEP1↓
```

When INSTEP1 is started, You will be asked many questions, and if all goes well, just about everything in the first phase of the *AIPS* installation will be taken care of by the procedure. A log of activities is kept in INSTEP1.LOG in whatever ends up being your $AIPS_ROOT area. This can be useful in keeping track of what was done, and may help in debugging any problems you encounter.

If something goes wrong, or if you interrupt it, INSTEP1 can be restarted. It checks if certain directories and/or environment variables are defined, and it reads the contents of the INSTEP1.STARTED file. However, it may repeat a few steps; this behaviour is in general harmless.

There are certain exceptions to this statement, *e.g.*, if you interrupt it after it has copied a template file such as PRDEVS.LIST to its destination but has not had a chance to finish work on it. Another example is when installing over an existing version; an interruption at the wrong moment can cause problems. The most common problem is when the $AIPS_ROOT scripts are not moved to *.OLD and replaced with the most recent versions.

The remedy depends on the problem (obviously!) In the case of a setup file like PRDEVS.LIST, you should edit the file by hand after INSTEP1 has finished. In the case of a more serious failure as above, where the $AIPS_ROOT scripts were not updated, removing INSTEP1.STARTED and restarting the script turns out to be the best solution. This forces it to start over and make no assumptions about what is already in place.

If you plan to have *AIPS* running on more than one machine, you should run the INSTEP1 procedure on *one host of each architecture* in turn (from the same directory, i.e. the $AIPS_ROOT area as mentioned in the previous subsection). It is *NOT* necessary to run INSTEP1 on each host of the same architecture; the shell script SYSETUP will take care of necessary setup for the second and subsequent hosts instead. You should defer SYSETUP until you have run INSTEP1 on all architectures for your site (at least all compatible architectures), as SYSETUP will correctly handle all hosts, regardless of architecture, marked in the HOSTS.LIST file as being in the same SITE.

Solaris users, please note: AIPS looks at SunOS 4.x and SunOS 5.x as different architectures: SUN4 *and* SOL*, respectively; also Sparc Ultra systems can be classified either as* SOL *or* SUL. The latter distinction is purely to enable Ultra-specific optimization options in the compilation routines; SOL binaries will run fine on an Ultra, but SUL binaries may run faster. Likewise, the two architectures HP and HP2 represent different compiler options. The latter should be used on HP 9000 systems of model 780 or above.

If you attempt to install *AIPS* as root, you will get an error message saying that this is not a good idea (it really isn't; many of the system files installed will need to be accessed in read/write mode by *AIPS* users). The procedure will then quit immediately. Please install *AIPS* from a non-privileged account; the only time you need special privileges is when defining the internet services. Just about everything else can and should be done from a normal account.

5.3.1 Bourne Shells on Ultrix, OSF/1, ConvexOS

In the many shell scripts that actually start *AIPS* and its various pieces, there are features used in some places that are not available on all systems. In particular, the default bourne shells on the older DEC Ultrix and Convex Unix systems do not have the required features (shell functions), and the bourne shell on Dec Alpha OSF/1 is missing a feature used in `INSTEP1`. So at the very start of `INSTEP1`, if it detects that you are on one of these systems, it will attempt to convert the relevant scripts to either the System V shell (`sh5`) on Ultrix, or the Korn shell (`ksh`) on ConvexOS and OSF/1. You will then see a message like this:

```
Please restart INSTEP1 again so /bin/ksh can be used.
```

or, for OSF/1:

```
Please restart me, I just changed my shell to ksh...
```

While the procedure attempts to convert all relevant scripts, you may find others later on that also need to be converted. In that case, you can use the `sh-cvt` shell script created by `INSTEP1`. It is left in the `$AIPS_ROOT` area. In the worst possible case, you will have to run `sh-cvt` by hand on `INSTEP1` and then start it again. This script is not created for OSF/1, only for Convex and Ultrix systems.

5.3.2 Protection Mask

The "umask" or protection mask is a number (in octal) that controls the protection applied to newly created files. If you type `umask` at the Unix prompt, it displays this octal number. Each digit represents a set of protection bits to apply to the owner, group and other users respectively. Depending on your system, the command will show it as a two, three, or four digit octal number. The last (least significant) two digits are the important ones.

A umask of `002` (or `02` or `0002`) is common and allows the owner and group to read and write files, and others to read them. The other common setting is `022` which is identical except it does not allow group write access.

If you plan on setting up *AIPS* for use with multiple users all in the same group, you will want to specify a umask of `002`. If a single user or account is going to be the sole user of *AIPS*, then you will probably want to use `022`.

When you have decided what value is appropriate for your installation, enter YES* if the value shown is what you want. Otherwise enter NO and the procedure will prompt you for a value.

* You may enter any of: YES, yes, Y, or y to indicate an affirmative response. Likewise NO, no, N, or n for a negative one; this applies to all yes/no prompts from INSTEP1

5.3.3 Restarting and Older Versions

The way INSTEP1 discovers the existence of older versions is through the file INSTEP1.STARTED, or the presence of a 15mmmYY directory (for 15APR92 or later). If the former file exists, it will be moved to INSTEP1.STARTED.OLD and you will be asked if the SITE and $AIPS_ROOT definitions from your older version can be used for the current installation. If you answer yes, these values will be automatically taken from the older file, and the date of the older version will be remembered. In all cases, a new INSTEP1.STARTED file will be created in the current directory. This is used to store the site name, *AIPS* version date, and the $AIPS_ROOT directory.

5.3.4 Host Name

Normally, you will not be asked what the computer's host name is, unless it cannot find the uname or hostname commands in your PATH environment variable or a few other likely places (this is very unlikely). You should see, *e.g.*:

```
INSTEP1:  The name for this AIPS HOST is ORANGUTAN
```

for a machine called orangutan.cv.nrao.edu. If it asks you for your hostname, the domain part of the internet name, if any, will be removed (in the example above, the domain part is .cv.nrao.edu).

5.3.5 Site Name

If you have not already done so (*e.g.*, in a previous installation), you need to think of a suitable short *single word* name for your *AIPS* site. An *AIPS* "site" covers all the computers that will be sharing this installation, and can include several different types of computers (as long as they can share binary data, *i.e.*, they have the same floating point format and byte order†). At NRAO, we have site names like NRAOCV for our Charlottesville site, NRAOAOC for the Array Operations Center, and so on. This name will be used for some files and directories, so do not put spaces, tabs, commas, or other unusual characters in it.

If you have not previously specified the site name, INSTEP1 will ask:

```
INSTEP1: Please enter a name for your site.  This should be a SINGLE
INSTEP1: WORD that can be used as a directory name.  Example: at NRAO
INSTEP1: we use names such as NRAOCV, NRAOAOC, and the like.

INSTEP1: ===> Enter your site name:
```

The procedure will convert whatever you enter to uppercase. If you are not running INSTEP1 for the first time, it will retrieve the name of the site from the file INSTEP1.STARTED.

† The following are all big-endian and use IEEE floating format: SUN3, SUN4, SOL, SUL, IBM, HP, SGI. These are little-endian and use IEEE f.p.: DEC, ALPHA, LINUX, AXLINUX. Finally, CVEX is big-endian and may or may not use IEEE f.p. format

5.3.6 Architecture

This part of the script tries to figure out what sort of computer you are installing this on. In most cases it will guess correctly; extensive testing has been performed on most modern systems. Here is an example of what you may see:

```
INSTEP1: host ORANGUTAN seems to be a Intel PC running Linux
INSTEP1: If this is correct, enter YES.  If not, enter NO and
INSTEP1: you will be given a choice of systems for which AIPS
INSTEP1: has a set of canned shell scripts.

INSTEP1: ===> Is this correct? (no default response)
```

Here you enter `YES` if it guessed correctly. In the unlikely event that `INSTEP1` gets the wrong architecture, or if it says that it is a "system this script hasn't seen before (*OS name here*)" or "system without uname or any other clue!", you enter `NO` to the "is this correct?" question. Then it will show you a list of known architectures and ask you to choose one or enter a new one. If your system is not on this list, you need to refer to the *AIPS Unix Porting Reference* (despite its outdatedness) in addition to this document as some OS-specific (Z) routines need created by hand.

`INSTEP1` now creates a set of architecture-specific directories including your `$SYSLOCAL` area (`$AIPS_ROOT/15OCT99/$ARCH/SYSTEM/$SITE`) unless this has already been done. `$SYSLOCAL` is then populated with files from various places: all files from `$ARCH/SYSTEM` (*e.g.*, `$SYSIBM`), and any of seven critical files in `$SYSUNIX` that were not already copied. For Convex and DEC systems, some of these are run through `sh-cvt` to convert their shells to korn or sh5.

If `INSTEP1` has detected an older *AIPS* version, you will now see a note reminding you that you may want to compare `CCOPTS.SH` and `LDOPTS.SH` files from the old `$SYSLOCAL` area with the new ones, and also to check `FDEFAULT.SH` in `$SYSUNIX` also. If there are any modifications you made to these or other files for your previous installation, you should compare the files (`diff` or `sdiff` are useful), and watch for changes in the new versions.

5.3.7 Verifying the AIPS ROOT Area

You should already be in this area, but the script asks you again just in case (only the first time you run `INSTEP1`; it stores your answer in the file `INSTEP1.STARTED` for future use).

```
INSTEP1: The current directory is
INSTEP1:  /home/aips

INSTEP1: ===> Please confirm that this is the AIPS_ROOT area: (YES)
```

You should enter `YES`. As `INSTEP1` insists on being run from the area that will be the `$AIPS_ROOT` area to begin with, your answer should always be yes here. The only exception might be if the `/bin/pwd` program does not hide the automounter's `/tmp_mnt` or similar mount point for NFS mounted disks. `INSTEP1` will remove any leading `/tmp_mnt` from the output of `pwd`. If this fails (or you use something other than `/tmp_mnt`), you probably need to edit INSTEP1 around the start of "Stage 6" so that it gets it right. If this is not the first time you are running INSTEP1 for the 15OCT99 release, you will simply be asked to verify that the value for $AIPS_ROOT is correct.

You have to use the same $AIPS_ROOT directory name (referred to by the same name) on each host in your site that will run this installation of *AIPS*. However, you may also choose to have multiple *AIPS* logical "sites" sharing the same $AIPS_ROOT. This is most commonly used when you have incompatible architectures (*i.e.*, little endian and big endian), in which case different site names are used for the two in the one common HOSTS.LIST file (see below).

5.3.8 Get the Binaries via Anonymous FTP

Unless you have obtained a tape or CD distribution of *AIPS*, you will be asked during INSTEP1 if you want to retrieve binaries via *anonymous ftp* from NRAO's *AIPS* ftp server aips.nrao.edu over the internet. If you choose this option, and the ftp site appears reachable (this is automatically tested via the ping command by INSTEP1), you will be given a choice among the architectures mentioned above in section 5.1.3, "*Binaries via Anonymous FTP*", namely ALPHA, AXLINUX, HP2, LINUX, SGI, SOL, and SUL.

Please remember the sizes for the binaries (back about 6 pages) before blindly going ahead with the request. Also, consider the time of day (non-office hours for the US East Coast will result in faster transfers) and the bandwidth of your internet connection. Do not attempt it over a 14.4k or 28k baud modem!

If you are installing from a binary tape, make sure that you have unpacked the binaries (via the supplied gunzip program if you do not have it already on your system). If you do not unpack them, INSTEP1 will blindly assume that you have a source-only distribution. For installations where you have already got the binaries via FTP, the FTPGET script that INSTEP1 calls will automatically unpack the files via gunzip, either the version installed on your system or the version included with the binaries.

While every effort is made by the script to see if all files were copied successfully, there may be some files that did not or that were only partially copied. Watch carefully for error messages during the file transfer. If some minor files are missing, you can always suspend the INSTEP1 process (usually via CTRL-Z), get them by hand, and then restart INSTEP1; it should then proceed as if everything was successful.

5.3.9 Defining your Printers

The way in which printers are defined to the *AIPS* system is via a file called PRDEVS.LIST in the $NET0 area. This is very much like the other *.LIST files used to define tape drives, hostnames, and data disk areas. INSTEP1 will try to find out what printers are on your system, either through an old PRDEVS.SH file or through the /etc/printcap file found on some systems. If you have an existing PRDEVS.LIST file, INSTEP1 skips this section, assuming it has already been set up correctly.

If an old PRDEVS.SH file was converted, you should end up with a workable version of the $AIPS_ROOT/DA00/PRDEVS.LIST file. If you have the EDITOR environment variable defined, you are asked if you want to edit the file now; it is a good idea to do so and check that the definitions are correct.

If there is no old PRDEVS.SH file, but your system has /etc/printcap (as many systems do), the names of all printers will be extracted and presented to you; you can then choose to add some, all, or none of those found to the PRDEVS.LIST file. If there are more than 15, it will *not* iterate through the list! Instead it asks how many you want to define.

If your system does *not* have an /etc/printcap file, you will be asked how many printers you want to define for *AIPS*.

Either way, you will also be asked the following for each printer:

- The printer type. This will be one of:
  - ▷ PS for PostScript printers (assumed monochrome);
  - ▷ PS-CMYK for Colo(u)r PostScript printers (you can say "color" or "colour");
  - ▷ TEXT for plain text printers;
  - ▷ QMS for QUIC (QMS, Talaris) printers (*not* QMS PostScript printers!); and
  - ▷ PREVIEW for screen previewers such as ghostview*.
- The printer options. This should be a comma-separated list of options *with no spaces between the options*. Currently there are three options:
  - ▷ DUP for printers that print on both sides of the paper;
  - ▷ DEF for the default printer; and
  - ▷ BIG=nnnn to specify that this is the printer for "big" listings, *i.e.*, those in excess of nnnn lines. At NRAO, we use BIG=1000.
- A description of the printer. This is what users will see when choosing a printer at *AIPS* startup and also when a print job or plot is sent to that printer. **You MUST include a description for each printer** or else the startup scripts will fail.

When all the information has been obtained, you will be given the option of editing the resulting DA00/PRDEVS.LIST file if you have set your $EDITOR environment variable.

Finally, you are asked to select from Size A or A4 paper. The default is "A" (standard in the US, 8.5 × 11 inches, 21.59 × 27.94 cm), but if you select A4 (the European and Australian standard), the ZLPCL2 script in $SYSUNIX is automatically modified so that it calls the PostScript generating program with the correct option for this size paper.

The following table illustrates how the format of the resulting page and the font size vary with the number of lines that you define:

Lines (A)	Lines (A4)	Orientation	Font size
97	95	Portrait	6.8
61	58	Landscape	8.0
54	52	Landscape	9.0
52	47	Landscape	9.25

The first two columns show the number of lines per page for "A" format and "A4" format pages, respectively. The third column indicates what orientation F2PS chooses

* Includes arbitrary shell scripts in your SYSLOCAL area; this is a useful way for accessing unconventional or remote printers normally inaccessible to your system

if the number of lines is equal to or less than the indicated value, and the fourth column shows the font size (in points) of the Courier font used for printing. If you perform a binary installation and do not re-run `FILAIP`, the system parameter file supplied will be pre-configured for 97 lines per page.

The `15APR97` release of *AIPS* changed the way in which the printer information was stored in the environment and passed to `AIPS` tasks that perform printing. If you will be supporting versions of *AIPS* older than `15APR97`, you need to copy the `ZLASCL` and `ZLPCL2` scripts from `15OCT99` to the older `$SYSUNIX` areas.

5.3.10 Copying files to the AIPS ROOT area

`INSTEP1` will now copy a set of files from the `$SYSUNIX` area to `$AIPS_ROOT`, and then calls another script (`AIPSROOT.DEFINE`) that will set the aips root definition automatically in all these files. You do not have to edit them yourself — at least not for this purpose. (Aside: if you ever move your installation to another disk location, run this script to redefine the `$AIPS_ROOT` environment variable in all necessary scripts.)

If old files from a previous installation are found here, they will be moved to, *e.g.*, `AIPSPATH.SH.OLD` the first time `INSTEP1` is run. On subsequent runs, it assumes (and states so loudly!) that the files are from the `15OCT99` release. The `HOSTS.LIST` file *will not be overwritten* if it exists. Be careful; if this moving of files does *not* occur, this could cause subtle but debilitating incorrect behaviour in the scripts later (and it might be hard to trace). This failure to move the old shell scripts can occur if `INSTEP1` is interrupted or fails at the wrong place. If this happens and you notice it, remove the `INSTEP1.STARTED` file and restart `INSTEP1` to get it right.

The procedure then informs you of the need to use a different version of `TVDEVS.SH` if an older version of *AIPS* (one that you intend on keeping around!) predates `15JAN95`. It asks if you want to use this modified version (needed because of the format change). If you do not, it simply makes a symlink to the `$SYSUNIX` version of `TVDEVS.SH`.

Two symbolic links `aips` and `AIPS` are inserted in the `$SYSLOCAL` directory; these point to the `$AIPS_ROOT/START_AIPS` shell script. This makes it easier for people to run `AIPS` once they have the reference to `LOGIN.SH` or `LOGIN.CSH` in their `.profile` or `.login` files (as these `LOGIN` scripts in turn add `$SYSLOCAL` to the default `PATH`).

Finally, for Digital–Ultrix and Convex installations, the script will copy the version of `START_AIPS` from the `$AIPS_ROOT` area to the `$SYSLOCAL` area as file `AIPS`, convert its shell from bourne to either `sh5` for Ultrix or `ksh` for Convex, and create a symlink `aips` that points to it. This is necessary for sites with multiple architectures that include a Convex or Ultrix system, as well as others that may not have the `ksh` or `sh5` shells.

5.3.11 Setting the AIPS versions

This step is always carried out, even if you repeat the running of `INSTEP1`. It scans `AIPSPATH.SH` for the definitions of the variables `OLD`, `NEW`, and `TST`, tells you what it found and asks if you want to change them:

```
INSTEP1:  The AIPS versions are currently:

INSTEP1:       OLD   $AIPS_ROOT/15OCT99
```

```
INSTEP1:      NEW   $AIPS_ROOT/15OCT99
INSTEP1:      TST   $AIPS_ROOT/15OCT99

INSTEP1:  ===> Is this what you want?  (YES)
```

(the script may insert values for OLD and NEW that match an old version if it detected one earlier). Just enter YES (or press return) if you will only have one version, or if INSTEP1 guessed the older version(s) correctly. Otherwise you will need to enter exactly what the version directories should be. *You should not attempt to mix versions of AIPS prior to* 15APR92 *with* 15APR92 *or later; the directory structure is incompatible between the two.* When you have entered the revised definitions of the *AIPS* versions, they are automatically put in the AIPSPATH.* files for you.

5.3.12 Setting up the User Data Areas — Background

Since the 15APR92 release, the way in which *AIPS* user data areas (directories) or "disks" are handled is one that allows flexibility in configuration and in selection of desired disks at startup time. Whereas older versions defined the names of a few disks in the middle of an obscure shell script, they are now set in a file called DADEVS.LIST stored in the area $NET0 (this is $AIPS_ROOT/DA00). Also, some parameters formerly in the system parameter file are now found in the file NETSP in this same area. While INSTEP1 will try to set these up for you, some editing — if only to change the default TIMDEST values — will probably be necessary on your part after INSTEP1 completes. The format of these plain text files is explained in comments in each of them.

AIPS users now can choose at startup time which data areas they want to have accessible for the session. By convention, these data areas are named after the computer on which they are physically mounted, but this is not a firm requirement (it is, however, a strong recommendation). An environment variable $DATA_ROOT will usually be set to $AIPS_ROOT/DATA/, and this area can either contain the actual user data directories themselves or symbolic links pointing at the real data areas.

The selection of data areas at AIPS run-time involves comparing the name of the data area to either the hostname of the local system, or one of the words specified in the DA= option to the AIPS startup. For example, on host orangutan, typing this:

```
$ aips da=valen,vulcan↓
```

will cause comparisons with "VALEN" and "VULCAN" (uppercase) as well as "ORANGUTAN". The comparison criterion is that the hostname (either current host or one specified in the DA= directive) must match a component in the data area pathname that is terminated either by the end of line, or by an underscore. **NOTE**: this is different from versions prior to 15APR98.

So, in the above example, a data area /DATA/ORANGUTAN will be selected, as would /DATA/VALEN_1, but /DATA/VULCANOID would not ("VULCAN" is in the string but not terminated by end of line or underscore).

The order in which data areas are chosen is specified as follows. There are three categories: local disks, required disks, and optional disks. Local disks are those whose pathname contains a match (as defined above) for the current hostname, such as ORANGUTAN in the example in the previous paragraph. Required disks are those listed with a "+" sign in the DADEVS.LIST file, and optional disks are any matching a request on the command line in the DA= directive. Within each of these categories, the data areas will be added to the list AIPS actually uses in the order they are found in the DADEVS.LIST file.

If you have implemented an NFS automounter (including amd), it is possible to set up an indirect map (auto.DATA is what NRAO uses) so that, *e.g.*, the second data area from a host called tarsier will be accessible as /DATA/TARSIER_2 anywhere in your *AIPS* network. This is how the installations at NRAO's Charlottesville and Socorro sites have been implemented. If you do not have a large site, the use of symbolic links will probably be a lot simpler. Regardless of how you implement the data areas, the INSTEP1 script will assume that you want to use this same naming convention for each data directory or symbolic link, *i.e.*, $DATA_ROOT/$HOST_*n*, where *n* is 1, 2, 3, *etc.* for each disk on computer $HOST. You can override this but then you may defeat the disk selection method in the *AIPS* startup scripts.

One final point on data areas: if you use symbolic links, the data directories themselves can have any name you like, but it is recommended that you retain the naming convention outlined in the previous paragraph for the symbolic links.

5.3.13 Setting up the User Data Areas — Details

What INSTEP1 will ask you is the name of the $DATA_ROOT directory. If you are re-running INSTEP1 or have an older *AIPS* installation, it will find the existing DA-DEVS.LIST file and attempt to extract the $DATA_ROOT from it.

In addition to setting up data areas, there is a separate *FITS* area set up by default as $AIPS_ROOT/FITS. Its purpose is as a temporary staging area for *FITS* disk files prior to reading them in to *AIPS* or after writing them out from *AIPS*. The script will detect if this area already exists and leave it (also if the FITS environment variable is defined). Otherwise it offers to make it a symbolic link to another area, or if you decline, create the actual directory. (*FITS* is the Flexible Image Transport System; see http://fits.cv.nrao.edu/ for additional details).

Next it will ask you how many user data areas on this host you want (up to 35 per host; max of 35 will be used per *AIPS* session). Then it will add that many entries to the DADEVS.LIST file, adding similar entries with generic TIMDEST (time destroy; see the AIPS help on it once you have things working) parameters to the NETSP file and finally will attempt to create a SPACE file in each directory. Suppose you entered this:

```
INSTEP1:  ===> How many AIPS data areas on YOURCPU do you want?  1↓
INSTEP1:  ===> will disks here be required regardless of host?
                 (NO)  yes↓
INSTEP1:  YOURCPU disk 1 will be /home/aips/DATA/YOURCPU_1
INSTEP1:  ===> Should I make it a symbolic link to another area?
                 (YES)  yes↓
INSTEP1:  ===> What directory should symlink
                 /home/aips/DATA/YOURCPU_1 point at?  /u/DA01↓
```

If you choose not to use symbolic links, the script allows you to specify any directory name, and if the directory does not exist, you have the option of having INSTEP1 create it for you.

If you do not create any directories or symbolic links, *you must do so before starting* AIPS *for the first time.* The AIPS program needs at least one user data area, as do some of the setup programs such as FILAIP. In order to be a valid *AIPS* user data area, there has to be an empty file in the relevant directory called SPACE. This can be done with the Unix command `touch SPACE` when you are in the directory.

The "required" question allows you to designate a certain host's disks as required no matter what disks the user requests in the *AIPS* startup; this can be useful as then the user's save/get and message files will always be in the same place (disk 1). Otherwise (with no required disks) the users will have to contend with different save/get and message files depending on how they start up AIPS (they may not like this). The disadvantage of this scheme is the slower speed with which messages will be written (the user's message file is always on the first disk) if disk 1 is accessed via NFS. On otherwise fast machines, this can be a significant impediment (in fact, NRAO has given up on the centralized "disk 1" model completely; the convenience was not worth the large performance hit it entailed).

If the data area's name matches the host's name, it is implicitly "required", even if marked as optional in the file.

You may want to edit the DADEVS.LIST and NETSP files later, to change the order of disks in the former, or the TIMDEST and/or reserved user numbers in the latter. It is the order the data areas appear in the DADEVS.LIST file that determines the order you get at run-time. If you have the EDITOR environment variable defined, you will be offered a chance to edit both of these files now.

The format for the DADEVS.LIST file is quite simple; either a plus or minus in column one depending on whether the area is required or optional, *exactly two spaces*, and the full pathname to the directory or symbolic link. The NETSP file is more complex but has extensive comments.

The TIMDEST and "allowed user" parameters in the NETSP file will override whatever is set using RUN SETPAR (see below). The former is used by the verb of the same name (see the help file $HLPFIL/TIMDEST.HLP or type HELP TIMDEST from within AIPS) and the latter can be used to restrict a disk to a set of up to 8 user numbers. All zero means no restriction, and a usernumber of -1 in the first slot means the disk is only for scratch files. **DO NOT PUT ANY TABS (control-I) IN THE NETSP FILE; IT WILL CAUSE AIPS TO BREAK**. You should also preserve the spacing between fields, though this is not required. Finally, there is no need to include user number 1 on any given data area, as this user (the *AIPS* manager) will automatically have access to all disks.

If you intend on having mixed "endian" architectures share the same $AIPS_ROOT area (*e.g.*, Sun and Alpha, or SGI and Linux), an added feature of the data area selection may interest you. If your first site is named NRAOCV, and your second VCOARN (as ours is), then you can create a file in $NET0 called DADEVS.LIST.VCOARN that will be used only by systems in that site. This is useful for segregating little-endian data areas from big-endian ones; the only alternative in this instance would be for the minority endian site to have individual host-specific DADEVS.LIST files in each host-specific DA00 area.

Finally, if your users want the freedom to customize the list of data areas themselves, they can copy the system DADEVS.LIST file to $HOME/.dadevs and edit it to suit themselves. The onus is on them to maintain it and track your (the *AIPS* Manager's) changes in the master version.

There is a limit of 512 *AIPS* data areas or disks in the NETSP file. This is imposed by the code in $APLGEN/ZDCHIN.FOR and if you need to increase it, you should edit that file before INSTEP2 (and let us know; you may be a record holder!). If you have a binary distribution and need to make this change, a complete rebuild of all tasks will also be necessary (*i.e.*, INSTEP4).

5.3.14 Tape Drives

Definitions for tape drives are stored in the file $NET0/TPDEVS.LIST. The INSTEP1 script will allow you to enter definitions and descriptions in this file in a question-and-answer session, but you can also edit it later and easily add to it as it's just clear text.

If the $NET0/TPDEVS.LIST file does not exist, or if it does exist but does not contain any definitions for the computer on which you are performing the installation, you will be asked if you want to define any tape drives for it. If you choose yes, the script will give a few lines of advice customized to your architecture, e.g. for SOL:

```
INSTEP1:  Device names for SunOS 5.x are /dev/rmt/<#><d><n>
INSTEP1:  where <#> is the device number 0, 1, etc., <d> is the
INSTEP1:  density, one of l, m, or h for low, medium, high; and
INSTEP1:  n (literally) signifies norewind.  DO NOT USE THE
INSTEP1:  BERKELEY OPTION (a trailing 'b').  AIPS will place
INSTEP1:  the appropriate density setting in the device name
INSTEP1:  when it is run, internally.
INSTEP1:  Possible tape drives found:  /dev/rmt/0ln /dev/rmt/1ln
```

IMPORTANT: read and follow the advice on device names, or NONE of the tape tasks may work!

The script will attempt to list all tape devices in the last line of this information. However, these may or may not be real tape drives, or they may not be suitable for *AIPS* (*e.g.*, QIC drives are unusable for *AIPS* as they cannot backspace over files). This attempt will be made for SUN3, SUN4, SOL, SUL, IBM, Dec-Ultrix, Dec-Alpha, and Linux (Intel and Alpha). For others (HP, SGI, and Convex), it will simply recommend what are likely device names. If you are unsure about the exact form of the device name, check with your local systems manager or administrator (and be prepared to show them the source code for ZMOUN2 for your architecture). Also, look in the shipped TPDEVS.LIST in $SYSUNIX as it has many examples of device names for different systems at NRAO. As a last resort, contact the *AIPS* helpline aipsmail@nrao.edu.

Regardless of the architecture, it is *critical* that you use the no-rewind form of the tape device name. For most Berkeley-like systems the letter "n" is prepended to part of the device name, *e.g.*, /dev/nrst0 on SunOS 4. For all systems except Linux, please remember that **the *AIPS* tape mounting software will adjust the device name for density at MOUNT time**; do *not* define two *AIPS* tape drives that point at the same physical device with different densities! Also you should avoid device names that force compression to be used on write, as there are no real standards for this and your data tapes written with hardware compression may be unreadable elsewhere.

DO NOT USE THE BERKELEY TYPE DEVICES UNDER SUNOS VERSION 5!!! These are typically /dev/rmt/0lnb (*i.e.*, a b is appended to the name). If you do, the tape routines will **skip every other file on tapes**.

Be careful when adding third party drives to Digital Unix systems; NRAO has found that for high density exabyte drives with compression, it is necessary to declare it as a DEC TKZ09 when setting up the system. This is so that when AIPS uses the different device names, the system will actually send the device the right commands to change density. Failure to do this resulted in the drive always writing compressed (and non-portable) data.

If you have an SGI system at Irix version 5 and are building *AIPS* from source, the INSTEP1 procedure will automatically enable the correct versions of the Z routines (there are different versions for Irix 5 and 6). The binaries shipped with the 15OCT99 version are for **Irix V6 only**.

It is important to pay attention to the advice as presented by INSTEP1 for tape devices. The *AIPS* "Z" routines that handle tape mounting (ZMOUN2.C usually, in areas like $APLSOL and $APLLINUX) expect a certain device name, and if you specify one that is very different from this expectation, your tape access from AIPS *will not work*.

If you choose a naming scheme different from the one suggested by INSTEP1, you will almost certainly have to modify the architecture-specific version of ZMOUN2.C so that it will change the name correctly for different densities, if the drive in question supports multiple densities.

After this list is presented, you will be asked how many tape drives to define for this host. Following this, you will be prompted for the device name for each tape drive you want to configure, and a description of it. The description will be what the *AIPS* users see each time they start up with local access to the particular tape drive. The script will try to check if the filename specified for the device is a character-special device; this may or may not work depending on your architecture and you may get an incorrect warning; you can probably ignore it safely if you are sure of the tape device name.

5.3.15 TPHOSTS: Securing access to your tapes and disks

AIPS provides a simple but effective authentication method for enabling remote hosts to access your local tape drives and FITS (and other) disk areas. The file $NET0/TPHOSTS is consulted by the TPMON daemons, and is set up by INSTEP1. It is a simple text file that has a line or lines indicating which hosts you wish to permit to have access to the local tape drives and FITS disk areas in your installation. For example, to allow all computers in the Astronomy Department at Zathras University, and one from the Physics Department, you might insert a couple of lines:

```
*.astro.zathras.edu
uppie.physics.zathras.edu
```

in the file, assuming they was the right address. For each line, you can specify either a single wildcard * at the beginning of an internet name as shown above, or at the end of an IP address, e.g. 192.33.115.*. You can specify up to 512 lines in the TPHOSTS file (comments or blank lines don't count). INSTEP1 will now semi-automatically set this file up for you.

WARNING: *Failure to set this file up correctly may expose your AIPS system to unauthorized users*! An inspection of the file is really essential after INSTEP1 concludes. Read the comments in the file for more details.

5.3.16 Figuring out Compilers, Making the Preprocessor

If you have an Intel/Linux system running an old version of the OS, you may need a patched version of ld. This only applies if you are using a.out format object files. Most newer versions (1.3 and up) use ELF format; if yours does, ignore this next paragraph.

The bug is a failure of ld to cope with aggregate arrays larger than 16 megabytes. A patched version created for 1.2.8 is included in your *AIPS* distribution and, if INSTEP1 thinks you need it, you will be asked if you want to unpack it. If you deem it necessary (certain tasks do not compile at all with the unpatched ld), you should replace /bin/ld with this newer version. Have your sysadmin do this (and check it over first). **NOTE**: the code to detect ELF support is not necessarily reliable in INSTEP1; don't assume it knows better than you.

For systems with the RPM for the older egcs 1.0.1 distributed by NRAO, where the g77 and gcc compilers are placed in /usr/egcs/bin, the INSTEP1 and other scripts will automatically use them. If they are installed elsewhere, you will have to specify the location at this point. It is **strongly** recommended that the egcs 1.1 compilers be used on Linux (Intel and Alpha) systems; they result in programs that are twice as fast as their counterparts compiled with gcc and f2c (as was done for all older *AIPS* versions).

Now the procedure tries to figure out the names for your Fortran and C language compilers, so it can start to build the utility programs. First it extracts the compiler names from $SYSUNIX/FDEFAULT.SH (Fortran), and $SYSLOCAL/CCOPTS.SH (C). If you are on a (pre-Solaris) Sun and not using the ANSI C compiler acc, it warns you and suggests that you use it if you have it (you can choose to use it at this point). It will ask you to confirm that these are the compilers you wanted, *e.g.*:

```
INSTEP1:  Fortran compiler is f77, C compiler is cc
INSTEP1:  ===> If this is correct, please enter YES:
```

If you are performing a binary installation, it also prints:

```
INSTEP1:  (As this is a binary installation, you may not need
INSTEP1:  these; please humor me anyway...)
```

This reflects the fact that you may not need to use the compilers at all in a binary installation. However, it is strongly recommended that you enter the information on valid compilers correctly, so that you can easily apply patches which may involve a recompile or relink.

If you answer NO to the above question, you will be asked for the names of both the Fortran and C compilers. You can either enter the exact path (*e.g.*, `/usr/lang/f77` or `/opt/SUNWspro/bin/f77` or `/usr/local/egcs/bin/g77`), or just the simple compiler name (*e.g.*, `xlf` or `gcc`). Only use the simple names if you are sure that these compilers are in the default `PATH`. In either case, `INSTEP1` checks if the compilers exist, and notifies you if they are not found. For non-binary installations, if either of the compilers is not found, the procedure halts with an error message.

On IBM RS/6000 and HP-UX systems, you do *not* want to use the `xlc` or `c89` compilers; use the default `cc` instead, or use GNU C (known to work under HP-UX). For CD-ROM or binary installations, this is not a concern as `NEWEST` and the other utility programs in `$SYSLOCAL` are available as binaries on the tape or via anonymous ftp; see below. Also, if you use GNU C, you may have to include `-lgnu` (or possibly `-lgcc`) in the `LINK` command specified in `$SYSLOCAL/LDOPTS.SH` as otherwise you can get an unresolved reference to `___eprintf` during `INSTEP3` (this may be a problem with your GCC installation).

If you change the name of the C compiler used, it will be updated in `CCOPTS.SH` automatically. Likewise, if the Fortran compiler is changed, `LDOPTS.SH` will be automatically updated. However, `$SYSUNIX/FDEFAULT.SH` has to be changed by hand. If you change the name of the Fortran compiler for `INSTEP1`, you will be given the option of editing this file now if you have defined the `EDITOR` environment variable. This file is divided up into architecture-specific sections and you should modify the section that is marked in comments as specific to your architecture.

Once you have specified the compilers, the script proceeds. For CD-ROM and binary installations, the pre-made versions of the programs `NEWEST`, `F2PS`, `F2TEXT`, `PRINTENV`, `PWD`, `REVENV`, and the preprocessor `PP.EXE` will be copied to the `$SYSLOCAL` area now. The `F2PS` and `F2TEXT` programs are simple filters that translate formatted FORTRAN output to either PostScript or plain text. The `PRINTENV` program is a precaution against some System V flavors of Unix that do not have a `printenv` command, and also for those where the built-in size of the environment in the system-supplied `printenv` is too small for all the *AIPS* environment variables. `PWD` is a utility program that simply prints the current working directory, minus most "funny" automounter mount points. **Be careful!** It will not work for all cases where you might have used symbolic links to split up, *e.g.*, architecture specific areas if you do not have enough space on one filesystem or partition. Finally, `REVENV` is a reverse environment variable name lookup program that will be used in a future version of *AIPS*.

For source installations, it will be necessary to compile and link these utility programs. For all except the preprocessor, this should be a simple operation and INSTEP1 will succeed in building the programs. However, for the *AIPS* Fortran preprocessor, the rebuild is very dependent on details of each architecture. On very old Linux, Cray, HP and IBM systems, special actions may be taken by INSTEP1. Refer to the source of the script for the details.

5.3.17 Shared and Debug Libraries

If you are using a CD-ROM or binary installation tape, you can safely skip this section. All *AIPS* binaries supplied by NRAO use static *AIPS* libraries.

On four or more architectures — the very old Sun3 and Sun4, and Solaris and HP — it is possible to build the AIPS libraries as dynamic or "shared". However, the problems associated with this option make it in our opinion more trouble than the somewhat dubious benefits. In fact INSTEP1 no longer even mentions this possibility, and will simply build static *AIPS* libraries in the $LIBR area.

While there may be some savings in disk space because of reduced size in the binaries, this will be greatly offset by the requirements of significantly increased swap and memory. If the resources are unavailable at run time, errors (bus error, others) will result which will confuse and frustrate your users.

If you really feel the need to use shared libraries, the relevant sections of INSTEP1 may be found, uncommented, and used. See also the commented out sections of INSTALL.TEX. Caveat emptor.

In addition, you can choose to generate two sets of libraries: one optimized and one for debugging. This feature is useful for NRAO and for sites who plan on doing any development or coding with *AIPS*, but it is not necessary otherwise.

5.3.18 The HOSTS.LIST File—Background

This file is a cornerstone of the network-based system. It specifies all *AIPS* hosts in a network, their architecture, and whether they are full-blown *AIPS* processing systems or just TV servers or X-term-like displays. There are three categories of hosts you can enter in this file: full-blown *AIPS* hosts, TV servers, and TV displays. The first type can run AIPS and the XAS program and the other servers, the second is not permitted to run AIPS, but can run XAS and the servers (to be displayed anywhere you like via X11), while the third can only be the recipient of the display of the TV and other servers from an *AIPS* host or a TV server. Typically, systems with a monochrome display (or no display at all, *e.g.*, file or compute servers) are good candidates for TV servers, and x terminals or systems on which you have not installed *AIPS* can make good TV displays.

In older versions (pre-15OCT98), there was a limit of one instance of the XAS program at a time per *AIPS* host or "TV server". This is no longer the case. The ability to use Unix sockets instead of INET sockets for communication between XAS and the *AIPS* tasks has now been built in to the startup scripts. If you specify aips tv=local on startup, you will get a Unix-socket-based TV. Up to 35 of these may be run on a single host (given enough memory! This many would require close to 500 Megabytes of RAM) and can be displayed on different X terminals, servers or workstations.

If you want to have the ability to run AIPS on one system and XAS on another, you should resort to the older INET socket based scheme (don't type tv=local). While multiple AIPS sessions on a single host can all communicate with a single "local" or Unix socket TV, this is not true for multiple AIPS sessions on remote hosts.

With the demise of the Image Catalog files on disk for the *AIPS* TV, the way in which remote displays are handled has become a lot easier. Suppose you visit a remote site, and sit down in front of the Linux system (vorlon) belonging to your collaborator. You log in to your Sparc Ultra (shadow) back home, but a grad student back home is already using *AIPS* and the XAS TV on it (shadow). As the Linux system (vorlon) has *AIPS* on it already (15OCT98 or later, same as your Ultra), you can start a local AIPS session on it to get the TV fired up. Then in your remote login to the Ultra, you start *AIPS* thus:

```
$ aips tv=vorlon.b5.org tvok↓
```

This tells *AIPS* on shadow that your TV is on the system indicated, and that it has already been started up. It will not interfere with the existing TV on shadow.

Versions of *AIPS* prior to 15JUL93 had a different format for the HOSTS.LIST file, and the TV information was set in the AIPSASSN.* files instead. If you intend on mixing 15OCT99 with a pre-15JUL93 version, make sure you use the newer format for HOSTS.LIST and the newer AIPSASSN.* files (you will want to move the old HOSTS.LIST out of the way *before* this installation). The newer AIPSASSN files are backwards compatible and will work with the older version.

5.3.19 Generating the HOSTS.LIST File

The HOSTS.LIST file resides in the $AIPS_ROOT directory. The INSTEP1 script will set it up for you so that you should not have to edit it yourself (except maybe to add a new host later). If you are repeating INSTEP1, or if your HOSTS.LIST file from an older *AIPS* installation is found, the script should find the definition for your host in the file. Otherwise, it will prompt you for a description:

INSTEP1: ===> ASTRO Description: *Public Workstation, room 217*↓

In this example, a line describing host ASTRO will be added to HOSTS.LIST. You will be asked if you want to add entries for other computers of the same architecture as your current host at this point.

The script will also ask you if you want to set up any TV servers or TV display hosts of the same architecture as the current host at this time. Remember that the TV servers you specify here should be the same architecture as your current host, but they will not be capable of running *AIPS*. There is no restriction on the architecture of the TV display systems, but you need to specify a default TV host for each TV display (you can actually use a TV server or an AIPS host for this purpose).

5.3.20 Making XAS

Before proceeding further, the script checks to see if you have a NIS ("yellow pages") map for internet services, and if not, if you have an `/etc/services` file. Whichever it finds first, it searches for the `sssin` service. If it finds the service in the NIS map, it states so. If not, and it finds it in the services file, it points out that you may have to edit `/etc/services` on other AIPS hosts in your network. Finally, if it's not defined in either, it warns that the TV will not work until the services map or file is edited. See below (section 7.5) for additional details. Remember, if you are updating from an older version, the `ssslock` service may not be defined yet. `INSTEP1` will check this also and remind you if necessary.

The rest of this section can be skipped if you are performing a CD-ROM or binary installation; the pre-built XAS binary should already be present in your `$LOAD` area.

If it is not, the script will try to compile and link ("make" in Unix terminology) the XAS TV server for X11-based workstations. The as-provided Makefile works with all the in-house AIPS systems NRAO operates without any changes, as it picks up the `$ARCH` environment variable and sets appropriate values.

It is strongly recommended that the GNU version of `make` (usually installed as `gmake`) be used to make XAS. If you have this utility on your system and accessible via your `PATH` environment variable, `INSTEP1` will find and use it.

If you choose to let `INSTEP1` build `XAS` on Sun or Solaris systems, and you do not have the `OPENWINHOME` environment variable defined, it attempts to define it for you by searching in more-or-less obvious places for the `openwin` directory. If it cannot find it, it will not attempt to build `XAS`. You can make `INSTEP1` use the MIT version of X11 by defining `OPENWINHOME` as, *e.g.*, `/usr/local/X11`, before starting `INSTEP1`.

Otherwise, it will move to the `$YSERV` area located in `15OCT99/Y/SERVERS`, make a new `XAS` directory there, and unpack the `XAS.SHR` archive in this new directory. For all systems, the utility program `UNSHR` will be built and run. This is then used to automatically unpack the archive.

On Linux systems, if you are using a 386 based system, you may want to remove `-O2 -m486` from the compiler options; these options are now the default, and are appropriate for Intel 486, Pentium, Pentium II, and Pentium Pro systems.

One very useful option that may be employed by `XAS` is use of the `MIT-SHM` shared memory extension. If you have `DISPLAY` set correctly, *and* the X11 program `xdpyinfo` exists in your `PATH`, *and* the display is the one on which the TV will run, this should work. The script will show this:

```
INSTEP1:  checking for MIT Shared Memory via /usr/X11/bin/xdpyinfo
INSTEP1:  Use MIT X11 shared memory extension:  YES
INSTEP1:  ===> Is this what you want?  (NO)
```

If the `xdpyinfo` program is not in your path, or the `DISPLAY` variable is not set, you merely get some advice on how to run the program. Then it asks you the same question as shown above.

On IBM RS/6000 systems, unless you or your systems administrator has rebuilt the X server, you will most likely not be able to use the shared memory option. Even then, we have been able to support shared memory under AIX 3.2.2 and 3.2.3 but not under 3.2.5. Almost all other X11 based servers (Sun, Alpha, SGI, HP, generic MIT X11, XFree86) have this extension.

The use of shared memory on Suns means that the System V kernel options should be configured. Check with your systems administrator to see if your kernel has the IPCSHMEM option included. If not, and if you want to have XAS use shared memory, you must have the kernel rebuilt to include this option.

Following this, the Makefile is edited automatically (to set the shared variable appropriately) and the "make" is started. This should end with xas being moved to the $LOAD area as XAS. Unlike older installations, the Makefile is now generic enough to work with little editing or intervention, having encoded in it the standard locations for X11 libraries and include files on all the target systems used during test installation prior to the release of this version.

The default make command on HP (HP-UX 9) and older SGI systems cannot handle the advanced technique used in the Makefile. The script warns you about this, but will not stop the make attempt. To modify the system so that it will build on these systems, move to the $YSERV/XAS area, edit the Makefile (save the original!) and substitute the word "SGI" or "HP" for the word "$(ARCH)" as appropriate wherever it occurs.

5.3.21 Deploying XHELP on your web server

A relatively new feature, introduced in the 15APR99 version, is the ability to specify a local (to your institution) web server for use in providing the web-based XHELP facility. A pair of scripts, namely ZXHLP2 in $SYSUNIX, and ZXHLP2.PL in $SYSPERL, make this facility possible. The former calls Netscape (or another browser) to in turn call the perl script through NRAO's web server in Charlottesville to generate the help page in HTML, with embedded links to other help pages scattered throughout the text. It is triggered when the user types XHELP instead of HELP within AIPS.

It is far preferable to run the perl script from your local web server than to go all the way to our (not very powerful) server and have to endure the world wide wait before seeing the help page in your browser. So we request that anyone installing *AIPS* with access to the CGI area of their local web server please consider installing the ZXHLP2.PL script therein, and modifying ZXHLP2 to point at it instead of us.

One requirement of a local version of this facility is that the web server have read access to the help files. If this is not possible, the facility cannot function. It needs to read the 15OCT99/HELP directory and files therein. Normally these could be made available via NFS mounts or automounts.

When you are doing the install, you will be asked if you want to change the default browser from Netscape. If you do, you will have to alter ZXHLP2 as the syntax for calling a different browser will surely not be the same. See the last line of the script.

Then, it will ask if you want to use a local web server for XHELP. If you choose to do this, you will be asked for two things:

- The disk location of your CGI area, *e.g.*, /home/www/cgi-bin.
- The URL this corresponds to, *e.g.*, http://www.minbar.edu/cgi-bin/.

INSTEP1 will then attempt to create a version of ZXHLP2.PL in the cgi-bin directory. If you don't have write access to this area, you should postpone setting up XHELP till afterwards and edit it and $SYSUNIX/ZXHLP2 by hand (it's not hard). If successful, it will offer to let you edit the file (it says FDEFAULT.SH but it means the perl script, really! Sorry...).

Then INSTEP1 will edit the ZXHELP shell script in $SYSUNIX, and make the necessary alterations. You should inspect both scripts to make sure the changes you requested are correct.

5.3.22 Building the GNU Readline Library

The GNU ReadLine system is packaged in the *AIPS* source distribution as four AIPS-SHAR files: READLINE.SHR, RLDOCS.SHR, RLSUPP.SHR and RLEXAMP.SHR. These are found in the $SYSUNIX area. INSTEP1 will attempt to unpack them and build the library for you if libreadline.a is not found in the $LIBR/GNU area. This only applies to from-source installations; the library should be present for CD-ROM and binary distributions. This "snapshot" of the readline library is version 2.2 taken for the 15OCT98 version; older versions may not work as well, and newer versions are as yet untried.

This installation is done in three parts: unpacking, configuration, and making. The first part uses the same UNSHR program as is used to unpack XAS.SHR (see above). If it has already been built, that version will be used. Otherwise it will be built in the $LIBR/GNU directory (where all the ReadLine work will be done).

NOTE: If you use symbolic links to split up your *AIPS* installation on several disks, the code in INSTEP1 that refers to the previously built version of UNSHR may fail, as it uses relative paths such as "../.." to find it.

Once a version of UNSHR is available, the four archives are unpacked. The success of this is tested on the presence of readline.c; if it is present, it is *assumed* the unpacking worked (likewise for install.sh and the RLSUPP.SHR archive). The status of the unpacking of the documentation and examples subdirectories is not checked, as they are not essential to the installation.

Now, the configure script is called (in "quiet" mode). This will take a little while to run, as it has to check all sorts of C language header files, how certain library calls work, and more. As this is a quite robust package that has been ported to many more systems than *AIPS* ever will be, you should not encounter any problems here. If configure is unsuccessful, you should run it by hand with no options (this uses a more verbose mode that will pinpoint the problems). In this unlikely case, move to $LIBR/GNU, set environment variable INSTALL to /bin/true, and type ./configure to start it.

Finally, once the configuration has been established, INSTEP1 will start the library building with the make command. On some systems, this can produce copious warnings (*e.g.*, Silicon Graphics), and on others it will be really quiet. But it will take some time. You will see the various "cc" commands scroll by, and hopefully at the end you will get the message that it "seemed to work".

5.4 To Register or Not

The purposes of Registering your *AIPS* installation are twofold. First, it is the *only means for obtaining software support from NRAO for AIPS*. Second, it enables NRAO to estimate the user base outside our own sites. If you are registered and are an Educational, Academic, or Research organization, you qualify for free support (on a "best effort" basis, i.e. no guarantee but we'll try our best!). If you are not registered, NRAO cannot offer you any help on *AIPS*, other than help getting registered. With increasingly limited resources, this is unfortunately necessary.

INSTEP1 will offer to register you at this point. If you have registered an older version of *AIPS*, it will detect this (via the file $AIPS_ROOT/REGISTER.INFO) and re-use what information it can. Registration of each new version is necessary; your old registration does not apply to the current version. If you choose to register, several questions will be asked, and some additional information pulled from the configuration files. The answers and information are then written in a somewhat terse but readable file ($AIPS_ROOT/REGISTER.INFO). If you can send electronic mail to Internet, the registration can be sent to us automatically via that route; otherwise you need to print out the specified file after entering the details and mail it to NRAO.

Should you register? Yes. If you are intent on using *AIPS* for Scientific Research, definitely. If your research is in Astronomy, absolutely! If it is in Radio Astronomy, *please* register! If in doubt, contact us and describe your situation as concisely as you can.

The data you send back with your registration is private and confidential, will not be disseminated to others outside NRAO without your express permission, and will only be used for purposes of helping you with your *AIPS* installation or for statistical purposes (*i.e.*, the distribution statistics published on our web pages).

If you do register, NRAO will send you back a set of 7 numeric keys; these should be entered into your *AIPS* system via the SETPAR program, and this process is described later in this document in section 7.3.4, *Registering your System with SETPAR and SETSP*.

5.5 Preparing for INSTEP2 — Initial Customization

This section can be skipped if you are performing a CD-ROM or binary installation, except for sub-sections marked with a "**".

The last thing INSTEP1 will do for you is to make an architecture-dependent installation directory 15OCT99/$ARCH/INSTALL and copy the other INSTEPn shell scripts there. Then it ends with a notice to edit certain files, check others, and finally move to this new install directory and start INSTEP2. In most of what follows, each section will begin with the messages you will see from INSTEP1.

5.5.1 Compiler Settings for Fortran and C

The options for the C compiler, assembler and linker are stored in four files in the $SYSLOCAL area: CCOPTS.SH for C-language, ASOPTS.SH for assembler (Convex only), LDOPTS.SH for the linker, and INCS.SH for alternate include areas. Move to the $SYS-LOCAL directory and check the *OPTS.SH files to see that your compiler options are set correctly. Most options should already be set for the supported architectures. You may want to set `PURGE=TRUE` to save space; this causes temporary log files, and intermediate pre-processed source files to be deleted after a successful compilation or link; this should be already set in most files by default. If you anticipate frequent recompiling or relinking of *AIPS* or other tasks, you should also make sure `SAVE=TRUE` is set in the `LDOPTS.SH` file; this avoids bus errors and other nasty behaviour when the binary file for a given program (or `AIPS` itself) is replaced by such a relink, and there happens to be an instance of the program active.

The Fortran compiler command and options are specified for all architectures in a single file `$SYSUNIX/FDEFAULT.SH`. In addition, the optimization and debug level used for different *AIPS* source code is set in `$SYSUNIX/OPTIMIZE.LIS`.

The `$SYSUNIX/FDEFAULT.SH` file contains compiler settings for Alpha (OSF1 and Linux), Convex, IBM, Sun (SunOS 4 and 5), Linux/Intel, DEC (DecFortran), HP and SGI compilers. It also has a very simple default for anything else (compiler is `fort77`, no debug or optimization or any other flags used). You should edit this file to insert definitions for your architecture if it's not one of the above (and also send them to us at `aipsmail@nrao.edu`, after you have verified they work!)

If your Convex has IEEE hardware, the `-fi` option needs to be added to the options files `FDEFAULT.SH` in `$SYSUNIX`, and `ASOPTS.SH`, `CCOPTS.SH`, and `LDOPTS.SH` in `$SYSLOCAL`. If you do not have the IEEE hardware, you need to make sure the `-fi` option is *not* in these files, and also edit `LIBR.DAT` to remove the reference to `$APLNRAO1` in the Z routine section at the top of the file. This latter action forces the use of the non-IEEE version of `ZDCHI2.FOR` which defines floating point formats for *AIPS*.

Some systems have an include area pointed to by environment variable, *e.g.*, `$INCALN`, which is referenced in the `INCS.SH` file. More likely the relevant area is just `$INC`. The `PAPC.INC` file therein defines the pseudo array processor size as 1.25 megawords (5 Megabytes) in most cases. Before starting `INSTEP2`, you should check if your version has an archicture-specific version of `PAPC.INC` (these do: ALN, CRI, CVEX, IBM). For systems with 128 or more megabytes of main memory, you may want to increase the AP size by editing the `PAPC.INC` file. As an example, on NRAO's large-memory IBM and Sun systems, the `APSIZE` variable in this file is set to 21233664 (81 megabytes). This is only useful for large scale problems, in this case images of size 4k on a side, to allow various *AIPS* tasks to read in the image entirely to memory, and tends to increase performance somewhat.

*5.5.2 ** Login or Profile Setup*

Before doing anything further, you need to do one more thing. It is important that the *AIPS* "logicals" (environment variables) for programming get defined before you proceed further, especially if you are performing a source installation. For binary or CD-ROM distributions, if you have the `EDITOR` environment variable set, you will be asked if you want to edit the relevant file now. It *should* guess your shell correctly; this worked in the installation testing where `csh`, `ksh`, and `bash` were used on various machines.

If you have the c-shell or tc-shell as your login shell, the line to insert in your `.login` file is:

```
source LOGIN.CSH; $CDTST
```

If your login shell is the BASH or Korn or Bourne shell, you will be editing the `.profile` file in your login area, and the relevant line is:

```
. ./LOGIN.SH; $CDTST
```

(Yes, that starts with "dot space dot slash").

The `$CDTST` is only really necessary for source installation, and/or if you plan on programming in *AIPS*. Otherwise just the reference to the `LOGIN.*` file should suffice in your startup file.

After editing whichever startup "dot" file is relevant for your shell, whenever you log in again you should see:

```
$AIPS_VERSION=/home/aips/15OCT99
```

(if you left in the `$CDTST`) where `/home/aips` is replaced with whatever you defined as your aips root area. If you type the above line for your shell interactively, you will also see this message.

*5.5.3 ** OPENWINHOME and TV Servers Script*

For any system where the environment variable `OPENWINHOME` is defined, (typically Suns and Linux systems, but Open Look has been ported to other systems), you will see a warning at the end of `INSTEP1` to "Check the definition of `OPENWINHOME` in `START_TVSERVERS`". If you are using OpenWindows as your X11 environment, you should do this and correct the definition for your site if necessary. If you are using vanilla X11 from MIT, or not using Suns or Open Look, the definition should not affect you.

5.6 ** Get any "Patches"

Regardless of whether you have a CD-ROM, binary or source-only installation, you need to check for the existence of any patches at this point. If you can, use the world wide web to check for patches at:

```
http://www.cv.nrao.edu/aips/15OCT99/patches.html
```

Alternatively, use anonymous ftp to `aips.nrao.edu` and retrieve (and read!) the file `/aips/15OCT99/README.15OCT99`.

These files will have detailed instructions on what files to copy, and where to put them. By getting the modified files now, you save having to recompile and/or relink them later.

If you are unable to use internet to get the patches, contact us via the means of your choice (see the title page of this document for alternatives). Likewise, if you obtained a CD-ROM or binary distribution and cannot rebuild because you don't have a Fortran compiler, contact us for help (we may not be able to do much in this latter case unless the circumstances are exceptional).

5.7 Start INSTEP2 and Additional Customizing

This section and its subsections can be skipped for the most part if you have performed the CD-ROM or binary installation process and do not plan on modifying any of the configuration files. It may be useful to read through some of it, however, as there are certain options that have to be enabled by hand if you want them. Pay special attention to the sections indicated by **.

5.7.1 Getting INSTEP2 Going

At this point, for source installations you are ready to start `INSTEP2`, and the remaining tasks to finish what `INSTEP1` started can be performed in parallel with it. The only common exception to this is new ports and Convex systems; see below. Otherwise, move to the install area and start it:

```
$ (INSTEP2 >/dev/null 2>INSTEP2.ERRS &)
```

This works for the bourne, korn or bash shells; if your shell is `csh` or `tcsh`, you should instead type:

```
% ( ( INSTEP2 >/dev/null ) >& INSTEP2.ERRS & )
```

(thanks to the many installers who pointed the above c-shell recipe out to me!) Then you can easily monitor progress with:

```
$ tail -f INSTEP2.LOG↓
```

Of course, if you prefer, you can dedicate your terminal or window to it and just type `INSTEP2`. The advantage of putting the process in the background is that you can then log off from that terminal or window and log back in later to check on things. The example shown above will also redirect any error messages (as opposed to informational ones) to a file `INSTEP2.ERRS`. This may contain some "tsort" cycle messages and possibly a notice about remaking `SEARCH0.DAT`, both of which can safely be ignored. One disadvantage of backgrounding the job is that it is very difficult to stop it (the author knows this from personal experience!) and doing so involves much use of `ps` and fast fingers.

Yet another alternative is to run as an `at` or `batch` job, for example:

```
$ at 8pm↓
at> INSTEP2 >/dev/null ↓
```

It is safe to redirect the output to the null device, as a copy of all output will be recorded in the `INSTEP2.LOG` file. If you prefer, you can leave out the redirection above and let the system mail you a copy of the output (this can get *very* long, however; maybe over a megabyte).

5.7.2 The LIBR.DAT file

If you are installing *AIPS* on any of the well-known systems, you will not need to edit the `LIBR.DAT` file at all; skip to the next section. The version now in your `$SYSLOCAL` area should be adequate. The only exception is for configuring TV's other than the standard TV server. The default Convex version of `LIBR.DAT` is a good example of one configured for three different types of TV. Comparing it to, *e.g.*, the Sun version shows the following features:

- The addition of two object libraries, one for `$LIBR/YM70/SUBLIB` and the other for `$LIBR/YIVAS/SUBLIB`.
- The library lists for each TV-dependent area like `$AIPPGM` is replicated twice, with fields `:2:` and `:4:` between the library name and the *AIPS* program area name.

If you only have one alternate TV, you should remove the sections for the IVAS and edit those for the IIS to match your alternate TV. Note also that the IVAS definitions have a NRAO-specific directory for the `XANTH.LIB` library; you need to replace this with the location of this IIS-supplied library on your system.

For more details on `LIBR.DAT`, or if you have to develop Z routines for your local site (unlikely unless you're doing a new port), you should refer to the (outdated but useful for mining) *AIPS Unix Porting Reference* for directions.

*5.7.3 ** Remote shell commands*

Finally, if you want the network system to *try* an automatic TV startups on remote logins, you *must* ensure that the `rsh` or `ssh` remote shell command works from each host in your system to all other *AIPS* hosts (on HP systems use `remsh` instead of `rsh`). Its purpose is to perform a single command on a remote host without performing a full login on that host.

Consult your systems or network expert on the use of the secure shell. The support within the *AIPS* shell scripts for alternatives to `rsh` is not completely stable, but if you set the environment variable `AIPSREMOTE` to, for example, `ssh`, then this will be used in place of `rsh` or `remsh` when trying to start up services remotely. You will need to make sure that the remote system has the same setting of `$AIPS_ROOT` as your local system. If not, do not attempt automatic remote TV startup; instead, start the TV locally and issue the `AIPS` command with the qualifier `tvok`. Also, the remote host should be in the secure shell's list of known hosts prior to any *AIPS* use of `ssh`, and if your passphrase is not blank, you need to have your identity known by the `ssh-agent` program (usually automatically started when you log in; use `ssh-add` to add your identity to it).

For the older, less secure remote shell command, a file called `.rhosts` (yes, it begins with a dot) will almost certainly need to be in the login area of any account used to run *AIPS*. Make this file owned by the account with protection so only that account can read and write it (see example below). Before doing this, make sure the `.rhosts` file does not already exist. If it does not, create one something like this:

```
$ cd↓
$ cat >>.rhosts↓
otherhost.myuniv.edu aips↓
CTRL-D        (press control-d or whatever the End-Of-File key is)
$ chmod go-rwx .rhosts↓
```

In this example, it will now be possible for users with account name `aips` on system `otherhost` to perform remote shell commands to the system you are on. You need to put one line in for each *AIPS* host. You can check if it works by trying, *e.g.*, from the other host `rsh firsthost whoami`. For additional details, check the manual pages on `rhosts` (especially the part about using netgroups; this can make the file a lot simpler) or contact your local network expert.

`INSTEP1` will not mention it, but there is a variable in the `START_AIPS` shell script that will be found in your `$AIPS_ROOT` area. This variable, `REMOTE_ROOT`, defines what the value of `$AIPS_ROOT` will be on the far end of any remote shell command to start something on another system (*e.g.*, TV servers, tape dæmons, *etc.*). You will need to change this value if your `$AIPS_ROOT` area is something other than "`/AIPS`".

Before creating, editing or removing a `.rhosts` file or moving over to the secure shell, contact your local network and/or security expert for details on the use of these files. Know and understand how this is going to affect security for your account before doing it!!

6. INSTEP2 details

This section is only relevant for source-only distributions. Skip ahead to INSTEP3 if you have a CD-ROM or binary installation.

You should have already checked the compiler and linker options files (see previous sections) and, if you are not using an *AIPS* TV server as your default TV and have hard-wired TVs, modified the LIBR.DAT file. You should also have started the procedure as outlined in the section called *Getting INSTEP2 going*.

Some older (quite a bit older) releases had problems with running INSTEP2 from the korn and tcsh shells, but this *should* not be the case anymore. However, if in doubt, you can change to another one like the bourne or bash shell (just type sh↓ or bash) *before* starting INSTEP2.

The compiling involves:

- Pre-process the files *.FOR and *.C into valid Fortran 77 and C source files *.f and *.c using the PP shell script and PP.EXE preprocessor;
- Compile the Fortran 77 files (or C source files) into object *.o files using the FC or AIPSCC shell scripts; and
- Move the object files to a library directory under $LIBR and, when all files in a given area are complete, make a library file out of the object files.

These steps are encapsulated in the COMRPL shell script, except for calling the LIBR script to make the object libraries, which is performed direcetly by INSTEP2.

6.1 How long? Did it work?

This procedure should run for hours compiling all the vast number of *AIPS* subroutines. On modern, fast systems it will take an hour or less. On older systems (such as a Sparcstation 2) it will take closer to 8 hours. Your experience will no doubt vary.

When INSTEP2 has finished, it should indicate that it ended successfully. If you want to be sure, type:

```
grep -v ^- *.LIS
find . -size 0 -print
```

The first command searches for any lines in the list files that do not have a minus in front of them, indicating bad compiles. If it prints any, you need to dig into the log file and find out why they failed. The second makes sure none of the files in the current directory (should be the installation area) are empty; if they are, something is wrong, perhaps you ran out of disk space (ignore any empty INSTEP2.ERRS files; these *should* be empty!).

6.2 What if it bombs on me?!

If a subroutine should fail to compile, INSTEP2 will stop after trying all other subroutines in that particular area. The end of the file `INSTEP2.LOG` will contain messages explaining the reasons for failure. For the standard architectures, this should not happen as extensive install testing has been performed prior to release. Often the reason is environmental, *e.g.*, running out of disk space or perhaps having a different compiler revision. The critical error message may have been sent to the `stderr` standard error output and may not appear in the log file, so you may want to check for this in the event `INSTEP2` stops.

If you have an idea how to fix the subroutine, move to the area where the source code is and copy the original:

```
$ cp FOO.FOR FOO.FOR.NRAO↓
$ chmod +w FOO.FOR↓
```

The second command makes it possible to edit the file. Use your favorite editor on `FOO.FOR`, and then try recompiling it:

```
$ COMRPL FOO.FOR↓
```

If you solve the problem, and you think it is a problem NRAO should know about, please send us the fixed file! (Preferably by e-mail to the address on the cover; check the patches first to see if we already have it covered). In any event, you can then either mark the file as done in the appropriate `.LIS` file (see below) and/or just restart `INSTEP2`.

If a compile failed and you want to just skip that file and continue, you must edit one of the list files in the `INSTALL` directory. `INSTEP2` determines which subroutines to compile from one of several lists here. The file names of the lists are:

AIPSUB.LIS	APLNOT.LIS	APLOOP.LIS	APLSUB.LIS
APLGEN.LIS	QSUB.LIS	QNOT.LIS	QDEV.LIS
QOOP.LIS	YSUB.LIS	YGEN.LIS	YNOT.LIS

One of these `.LIS` files will contain the name of the offending subroutine. Edit that file and put a "–" sign in front of the subroutine name. Then restart INSTEP2.

```
$ cd $AIPS_ROOT/15OCT99/$ARCH/INSTALL↓
$ INSTEP2↓
```

7. INSTEP3

If you have performed a CD-ROM or binary installation for this architecture, you do not need to actually run the INSTEP3 shell script. However, if you plan on running the programs that create the system files by hand, please read the subsection below on *Making the System Files*.

If you are installing the system on one of the well-known architectures and are confident enough, you can in fact skip to running INSTEP4. However, remember to come back to the "*Making the System files*" section below after it has finished.

7.1 Running INSTEP3

INSTEP3 is also almost completely automatic. All you have to do is move to the INSTALL directory (you should be there already) and start it:

```
$ INSTEP3↓
```

This step will not take long (about 30 minutes on a Sparc IPX) and when it finishes, should print an "Ends Successfully" message on the terminal and at the end of file INSTEP3.LOG. It compiles programs listed in the file INSTEP3.LIS in the same directory. This is a subset of the files in INSTEP4, just enough to run the DDT test (see the separate section on this below) and check that *AIPS* is indeed functioning correctly. If you are confident that everything will work, you can start INSTEP4 in its place now (though this will take longer).

7.2 Making the System files

If you have performed a CD-ROM or binary installation, INSTEP1 should already have provided the system files for you, or given you the chance to run the programs that make the system files directly. You should in general only need to refer to this section if you need to run the programs by hand (as you will in a source-only installation) or if you want to know a little more about what the programs actually do.

For source installations, you must "run" the three programs FILAIP, POPSGN and SETPAR to create files needed by *AIPS*:

```
$ RUN FILAIP↓
$ RUN POPSGN↓
$ RUN SETPAR↓
```

Note: You need to run at least the first two for source installations, even if you have already set up your system files from a previous *AIPS* release. The mix of files these programs generate and their sizes changed subtly in the 15OCT98 release in that there is no longer a need for most of the image catalog files (IC and ID) and the task data file has a new number ("TDD000004;"; version 4; version 1 used by older systems). The sizes of the memory files was substantially increased; if you have a version of *AIPS* predating 15APR99, you need to replace *all* your memory files with the new larger ones. They are backwards compatible, *i.e.*, older *AIPS* versions can use the new larger files. This version cannot use the older, smaller memory files though so you *must* replace them.

You will see the same Data disk assignments message as you do in starting up AIPS when using RUN on these programs.

7.2.1 Running FILAIP

If you have an existing installation, doing this will generate many warning messages about already existing files and files not initialized, but it *will* set up the version-specific memory file correctly, and will add the modified task data file version 4 which is not present in versions prior to 15APR98.

If you do not have an older version of *AIPS*, or your older version is 15OCT98 or newer, then your memory files are fine. If on the other hand your older version predates the 15OCT98 release, you *must* replace the memory files in the $DA00 areas. Now may be a good time to move or remove the old ones in $NET0/*/MED*, though make sure no-one is running *AIPS* before you do this. Do this via:

```
$ rm $NET0/*/MED*↓
```

Make sure nobody is running *AIPS* when you do this!

You will then need to continue with FILAIP and POPSGN (see below), and if you have more than one *AIPS* host, set up the TEMPLATE area again with the newly generated files and run the SYSETUP script.

When run, FILAIP asks for certain set-up parameters:

```
# disks, # cat entries/disk (<0 => private catlgs)
```

The recommended values are 35, -100. The maximum number of disks is 35 for any given *AIPS* session, and you should specify some **negative** number for "cat entries" to force the use of private catalogs; the alternative is to have one mammoth public catalog for all *AIPS* usernumbers. Catalogs are extensible so you do not need to worry about 100 being too few entries. Then it asks:

```
# interactive AIPS, # batch queues (2 I)
```

The first number is how many simultaneous *AIPS* users you wish to allow *on this host*. A typical number is 8. The second is how many *AIPS* batch queues you will allow (2 if you need them is a good value, use 0 if you don't want any). The third item FILAIP asks for is:

```
#tape drives (I)
```

This should be set initially to $2+n$, where n is the number of tape drives on whichever *AIPS* host in your network has the most drives. Do not count QIC (quarter-inch cartridge) drives; they cannot backspace over files and are unusable as *AIPS* tape drives. So if the system you are installing on has, *e.g.*, an exabyte drive, but another workstation that will run *AIPS* has two DAT tape drives, you should enter 4. The extra two are for remote tape access.

Those of you who have been through earlier *AIPS* installations will notice the absence of any reference to the TV's. This is because the image catalogs have been moved into the memory of the server itself. There is no need in 15APR98 and newer versions to set up any image catalog files, nor is there a need for the ID (TV lock) files with the new TV lock server (TVSERV). In fact, if you no longer have versions of *AIPS* that predate 15APR98, you can dispense with all the $NET0/*/IDD* files, and all but one of the corresponding ICD files (ICD000000;).

After this, FILAIP has all the info it needs. On Suns running SunOS 4 and 5, when FILAIP and all other *AIPS* tasks end, they will print out the rather scary message:

```
FILAI1: minbar 15OCT98 TST: Cpu=    0.05  Real=     1.0  IO=         1
 Note: Following IEEE floating-point traps enabled; see ieee_handler(3M):
 Overflow;  Division by Zero;  Invalid Operand;
 Sun's implementation of IEEE arithmetic is discussed in
 the Numerical Computation Guide.
```

Do not be alarmed by this. The second line says ... traps *enabled*, not *occured*, in other words, the system was set to look for overflow, divide by zero, etc. problems but none of these occured.

7.2.2 Running POPSGN

This task initializes the *AIPS* memory files. For source installations, it is essential that this be run so that the version-specific memory file can be correctly initialized. The FILAIP program will already have created this file in the correct area.

When INSTEP1 runs POPSGN, the input is obtained from a file, and its output is diverted there too. You will never see these temporary files unless something goes wrong. If you run it by hand, however, with the command RUN POPSGN, you will see:

```
Enter Idebug, Mname, Version (1 I, 2 A's) (NO COMMAS)
```

and you will enter "0 POPSDAT TST" (the first character is a zero). Then a short while later, you will see a ">" prompt appear. Press <return>. That is all you need to do.

If POPSGN fails under Linux (this was known to happen in the a.out version 1.2.8 for older versions of *AIPS*), try recompiling KWICK and MASSGN with COMRPL DEBUG NOOPT $AIPSUB/KWICK $AIPSUB/MASSGN and then COMLNK $AIPPGM/POPSGN.

7.2.3 Running SETPAR or SETSP

SETPAR has over 30 options that you can change. It is likely that the only parameters of interest are:

- **3** Number of lines per "crt" or screen page; 40 is a good value if your terminal windows are typically this long. You can now set the number of lines per CRT page to zero; this enables code that will allow *AIPS* to figure out the size of the xterm or other window in which you are running AIPS) and adjust this parameter dynamically. Changing the window size results in AIPS adjusting the parameter.
- **4** Number of lines per print page; see the table listed in the "*Defining your Printers*" section above, as what you set here will define whether your printouts are rendered in portrait or landscape mode.
- **19** System name; see section 7.3.4 below, "*Registering your System with SETPAR and SETSP*. Bear in mind that *AIPS* tasks only use the first 6 letters of the system name string in their termination message. Some tasks that perform printing use more of it.
- **35** Computer speed rating; set this to what you expect your system to get in terms of *AIPSmarks*. This can have a big impact on performance if your tasks run

to completion quickly as it affects the delays inherent between the task and the AIPS interpreter noticing the task has finished. Refer to the *AIPS* web pages (see cover page) under "benchmarking" for more details. Briefly, a Pentium II 450 should be set to 16, a Sparc Ultra 10 should use about 10, and an old Sparc IPX would use about 1.

In versions of *AIPS* prior to 15JAN94 it was necessary to set the pseudo-AP 2nd memory; this is no longer true. It now defaults to whatever you had set in the file PAPC.INC when SETPAR.FOR was built. It is possible to lower this value on individual machines, and this is a fairly good way of allowing the same installation of *AIPS* to take account of larger or smaller memories on different systems.

Remember that the TIMDEST and reserved user/disk parameters are now obtained from $NET0/NETSP; the values you see in the SETPAR output are only system defaults. The SETPAR values will get used for any data areas in any DADEVS.LIST or .dadevs file which does not have a corresponding entry in NETSP.

At the end, it prompts you for a password. The "as-shipped" default is AMANAGER, uppercase. We strongly recommend this be changed later from within AIPS (for user number 1).

There is another program, SETSP, that sites with many *AIPS* systems will find very useful. Basically, it allows you to examine and change most SETPAR parameters for all hosts in your *AIPS* network at one go. In versions prior to 15JAN95, a separate file $NET0/SPLIST was needed to specify the list of system parameter files. This is no longer the case; the host names are read directly from the HOSTS.LIST file now. As this program is only built in INSTEP4, if you want to use it now, you need to build and run it by typing:

```
$ COMLNK $AIPPGM/SETSP.FOR↓
(verbose output from COMLNK appears here)
$ RUN SETSP↓
```

Also, there is an extra option built into SETSP. You can choose to change a given parameter, and set it to a *different* value for each different host. This can be a considerable time saver for moderate to large installations.

7.2.4 Virtual and True Color Maps

Starting with the 15OCT98 release of *AIPS*, the XAS TV server has the ability to use either an 8-bit "pseudocolor" visual or a full 24-bit "truecolor" visual under the X11 windowing system (this includes Sun's OpenWindows, the CDE, and the XFree86 and competing commercial X Servers). The basic difference between these two "visuals" is that the pseudocolor one restricts you to 256 different colors at a time (splitting up the 8 available bits among the three primary colors), whereas the truecolor visual provides more simultaneous colors than can be shown at a time on most monitors (16.7 million). You will know which the TV is using by looking at its icon; it plainly indicates if a 24-bit TV is in use.

The good news in using a 24-bit TV is that now tasks like TVHUI and TVRGB can yield results that many of us have not seen in *AIPS* since the demise of the old IIS and IVAS systems. The bad news is that the X11 truecolor visual provides *no* lookup tables for the usual contrast/brightness sweep or adjustment, so XAS has to provide this functionality itself. Thus, all the image has to be refreshed each time you run TVFIDDLE or related tasks or adverbs and adjust the brightness or contrast. This makes these functions a little slower.

The default behaviour is to attempt to use a truecolor visual if one is available from your X server. You can change this by modifying the X resources database (usually with xrdb or through the .Xdefaults file in your home directory) to set:

```
AIPStv*useTrueColor:  0
```

There are other X resources as well, for example if you want to alter the standard number of "gray" levels (default 199) that XAS will use in an 8-bit pseudocolor visual:

```
AIPStv*maxGreyLevel:  236
```

The above is the maximum permissible value. Lower numbers still give acceptable image viewing and avoid the situation where the default X windows colormap does not have enough entries free for XAS to run without a *virtual colormap* (if it uses a virtual one, you get "flashing" when the focus moves to XAS and the color of every other window on your workstation changes). Of course, there is no flashing whatsoever if you are using a truecolor visual (24-bit XAS).

See the XAS on-line help within AIPS for more details; there are other X resources that allow you to alter things within the TV.

Finally, as already stated, the first thing you or any AIPS user should do when the TV server starts up for the first time is TVINIT from within AIPS. This synchronizes the AIPS program and the TV server, though it is not as important in recent versions as it was in older, disk-based-image-catalog versions of *AIPS*.

7.3 System Files for CD-ROM or Binary Installations

This section and all of its subsections should be skipped if you are performing an installation from source code only.

7.3.1 Characteristics of the Pre-Built System Files

Near its end, INSTEP1 will tell you about the pre-made system files that have been extracted from the tape. You will be shown the parameters used in creating them, and given a choice: use the files with these parameters, or re-run the setup programs with different parameters. For the 15OCT99 version, these parameters are:

```
# disks, # catalog entries per disk:  35 −100↓
# interactive AIPS, # batch queues:  8 2↓
# tape drives:  4↓
```

Be careful: on SunOS systems that do *not* have the Sun Fortran compiler installed, you cannot at this point have INSTEP1 run any of the *AIPS* binaries, including FILAIP and other utilities. Most likely the Fortran run-time shared library libF77.so is not accessible. You should add the location of this library (INSTEP1 puts it in $AIPS_ROOT) to the $LD_LIBRARY_PATH environment variable before proceeding further.

The first line shows that the maximum number of per-session "disks" or user data areas has been configured (max 35). There is little overhead in this and it is recommended that you do not change it. The value of −100 for the number of catalog entries is also standard and the negative value causes private catalogs will be used (based on *AIPS* user number) and the initial allotment of entries in the catalogs will be 100 (they automatically extend themselves as needed). *A positive number indicates a global catalog for all user numbers; we do NOT recommend you try this unless you know what you're doing.*

The second line indicates that the provided system files can support a maximum of four simultaneous interactive *AIPS* users on any one host (*i.e.*, 8 for two hosts, 12 for three, and so on). In addition, there are two *AIPS* batch queues per host configured. If you anticipate having more simultaneous users than this, or require more batch queues, you will need to re-generate the system files with `FILAIP`.

Finally, the third line shows the files are configured for up to 4 tape devices. Two of these are "remote" slots, so the files as shipped can support up to two physical tape devices per host. If you need more, you should choose to run `FILAIP` below.

If you have an existing *AIPS* installation that predates `15OCT98`, you will now be notified of the increase in size of the memory (`ME`) files that came with this version. It will then be necessary to replace the old memory files in all the host-specific `$NET0/*` directories. You will be asked if you want to do this now, and cautioned to make sure nobody is running `AIPS` at the time. The process leaves the old files as, *e.g.*, `MED000001;.OLD` in these areas, so you can revert if there are problems (there should not be) or simply remove the old files later. The list of hosts the procedure operates on comes from your `HOSTS.LIST` file.

If you have an old version of *AIPS* that predates `15JAN95`, a reminder of the `UPDAT` program for converting the format of user's data will be issued at this point.

7.3.2 Regenerate the System Files?

After reading the previous section, you should decide whether you want to use the provided system files, or generate your own. If you decide you want to change them, `INSTEP1` will run `FILAIP` and `POPSGN` for you.

For most installations, it is likely that the system files provided on the tape will be adequate. You should only have to re-generate the files if your system or network requirements exceed the parameters as outlined in the previous section. The most common reasons for requiring the files to be re-generated will be either that you need more TV devices or more interactive *AIPS* sessions.

If you do choose to re-generate the system files, any existing files in the directories `$AIPS_ROOT/DA00/$HOST/` and `$AIPS_ROOT/$VERS/$ARCH/MEMORY` are removed and then the `FILAIP` program is started. It will prompt you for the values discussed in the previous section. Unlike older versions, you will *not* be prompted for details about each TV (the image catalogs are now stored inside `XAS` itself).

Assuming that `FILAIP` runs successfully, the `POPSGN` program will be run next. Unlike `FILAIP` (and unlike the source-only installation), you will not have to enter any parameters for this program; `INSTEP1` enters them for you and diverts the output. Normally all you will see is:

```
INSTEP1:  Starting program POPSGN...
INSTEP1:  POPSGN appears to have completed with no errors
```

7.3.3 Setting up a Template Area and SYSETUP

While you still have "untouched" system files, you are offered the opportunity to copy some of them to a *template* area. This is *essential* for those sites with more than one computer of a given architecture, and is strongly recommended if you plan on adding more *AIPS* hosts to your installation in the future. Choosing to set up a template area will make it possible for you to run the SYSETUP script to configure your additional hosts not just at installation time, but also anytime you make a change in your configuration. **The template files are version dependent and should be set up again for each new version of *AIPS*.**

The extra disk space needed for the template area will depend on the parameters you enter in FILAIP. If you do not depart too much from the default parameters as described above, you will probably only need an extra 8 or 9 megabytes.

INSTEP1 will then ask you if you wish to run SYSETUP to configure any additional hosts or TV servers of this architecture. If you wish to do this, you *must* have already set up the template area as described above, and you should have inserted the additional host names in HOSTS.LIST. SYSETUP will fail completely if you have not done this.

Choosing to run SYSETUP will bring up three questions:

```
Enter the AIPS manager account name [aipsmgr]:
Enter the AIPS user group name [aipsuser]:
Enter master host name:
```

The values in square brackets are the default values for the answers. You should always enter the current account name for the first question (the person installing *AIPS* should be the *AIPS* manager!). If you have not set up a special group in /etc/group for *AIPS* users, then simply use the default group of your current account in response to the second question. Finally, you should enter this host's name for the third question. There is nothing really special about the "master" host; its system area is simply used as a convenience in the internals of SYSETUP.

The SYSETUP script will tell you exactly what it is doing as it works. It will copy certain files from the TEMPLATE area. It also creates many symbolic links the system areas of files other than the master host, so that, *e.g.*, there is in effect only one *AIPS* password file. Other files that are symbolically linked include the gripes file. With the remaining files (accounting, batch, memory, system parameters, task and tape control files), it is necessary for each host to have its own copy, hence they are copied from the template area.

7.3.4 Registering your System with SETPAR and SETSP

The last item performed by INSTEP1 for the CD-ROM or binary installation will be to offer to run SETPAR and SETSP. This program allows you to change many items in the system parameter (SP) file, and is also where you enter your registration keys. When you submit a registration to NRAO, you will get back (via email or regular mail, whichever you used) a set of seven rather long numeric keys. These need to be entered into SETPAR and once they are entered correctly, you will be able to set the *AIPS* system name. Unregistered copies will not be able to set this field, and it will show "UN-REGISTERED!!!" instead.

The script will ask if you want to run SETPAR, and if you choose to, it will give you these instructions:

```
INSTEP1: The default password (as shipped) is AMANAGER (uppercase).
INSTEP1: (you can change this later inside AIPS for user number 1)
```

Then it starts the SETPAR program. You will first see a menu asking you to choose a number; you should choose option 2 which allows you to change the parameters. Once you do this, the program shows you *all* the system parameters. Item 19 will be the "local system Registration" and its value will appear in various *AIPS* printouts for your users (mostly the first 6 characters, in some cases more). Enter 19 to choose that item. When you do, you will be asked to enter the six main keys in order; be careful to enter them correctly (a single digit misplaced will cause the registration to fail). Once they are all entered, you will be prompted for the version-specific code that you recieved from NRAO. Once this is entered, it will verify the registration, and if successful, ask you to enter the *AIPS* system name for this host. Up to three different versions of *AIPS* can be registered at a time per host.

To register the other systems in your site, you need to run SETSP, and INSTEP1 will do this for you if you ask it to. The message you will see if you do is:

```
INSTEP1: The default as-shipped password is AMANAGER
INSTEP1: Choose "19 CHANGE" once you are in the program
INSTEP1: and then preserve (via -1) the existing values for
INSTEP1: this host.  You will then be prompted for the system
INSTEP1: names for this and each other host.
```

Because you have already registered your first system for the 15OCT99 version of *AIPS*, the default values for the six keys — and the version-specific code — will be what you have already entered. SETSP allows you to preserve these by simply entering -1 for each of them, so this is what you should do. It will then ask you to re-enter the *AIPS* System Name (sorry, no defaulting done; I know this is silly). More importantly, it will now automatically register every other system in your site and you can enter their system names too.

When in SETPAR and SETSP, if you are asked for the password, remember that this is the *AIPS* password for user number 1 (AMANAGER as shipped, case sensitive) and this is *not* the account password! (You should change the *AIPS* password for user 1 as soon as you can, however).

7.3.5 Warning Messages: Message File, Internet Services

If you see messages about file MSD001000.001;, in particular something like "*File MSD001000.001; already exists, not initialized*" or "*File MSD001000.001; not found*", you can safely ignore them. This is referring to the message file for user number 1. If it already exists, FILAIP will not initialize it. The former occurs if you have an existing *AIPS* installation and are upgrading to the 15OCT99 version. If POPSGN or SETPAR complain about not finding it, the installation script may have failed to copy it to the "disk 1" area. This is not important as this message file will be created whenever user 1 enters AIPS for the first time.

Although you are now finished with the *AIPS* installation, you will still not be able to run AIPS without some rather ominous error messages at this point (though if you specify "aips notv tpok" on the command line, it should work). These will most likely involve the internet services, and you should refer to section 7.5 (*Setting up the Internet Services*) on this topic. That section shows you how to modify /etc/services or its yellow pages (or NIS) equivalent, the internet services map, so that *AIPS* can find the internet sockets it needs for the TV servers and Tape dæmon TPMON. What many people do not realize when they see all the messages that result when the services are not defined is that the core of AIPS itself is working.

Once you have these issues resolved, and have checked the various *.LIST files in $AIPS_ROOT and $NET0, you should be finished with your *AIPS* installation for this architecture.

If you have user data from a version of *AIPS* that predates 15JAN95, you now need to RUN UPDAT to convert user data from the old "C" format to the newer "D" format. See section 9 below, *UPDAT — CONVERTING OLD USER DATA* for the details. **Do not attempt this for users who have both "C" and "D" format data on the same disk**!

Other than this conversion, the remainder of this document is almost exclusively for source-only installations, though some of the items in the next and previous sections are relevant if you plan on changing or customizing certain aspects of your installation. Any section or subsection with two stars ** preceding the section name is relevant to CD-ROM, binary and source installations and should be read.

7.4 SYSETUP – Setting up System Areas for Other Hosts

As described in section 7.3.3 for CD-ROM or binary installations (*Setting up a Template Area and SYSETUP*), this script will allow you to set up the system (DA00) areas for a large number of hosts at one time. It assumes that you have already set up at least one system (using FILAIP and POPSGN). It refers to this system as the *master* host. If you are installing from source, you should refer back to that section now. If you are performing a binary or CD-ROM installation, skip the rest of this subsection.

To set up the TEMPLATE directory, you should do the following:

```
$ mkdir $AIPS_VERSION/$ARCH/TEMPLATE↓
$ cd $AIPS_VERSION/$ARCH/TEMPLATE↓
$ cp $AIPS_ROOT/DA00/$HOST/* .↓
```

This really should be done *right after* you have created and initialized the files for your first host (after POPSGN but before SETPAR). While this area is nominally architecture-dependent, it actually only depends on the $SITE, so you could in principle create the TEMPLATE areas for other architectures via symlinks to the first one you generate (but only for hosts within the same logical "site", *i.e.*, same byte order and same f.p. format). This will save a moderate amount of space.

Now you just type SYSETUP and sit back while it does all the work of moving files and setting up hard links. It keeps you informed on what it's doing. You can also use SYSETUP to add a new host by saying SYSETUP <hostname>.

7.5 ** Setting up the Internet Services: /etc/services

Or: **HELP! My TV server, Tek server, Message Server, or TPMON won't start!** And what are all these error messages???

Before any of these nice features of *AIPS* can work, you *must* have certain internet services defined. These are normally defined in the `/etc/services` file* which requires root (system administrator) privilege to change. If you do not have access to the root password, you should contact someone who does.

The simplest thing to do is to copy the text below directly into the end of the `/etc/services` file or its equivalent YP/NIS map:

```
sssin        5000/tcp     SSSIN      # AIPS TV server
ssslock      5002/tcp     SSSLOCK    # AIPS TV Lock
msgserv      5008/tcp     MSGSERV    # AIPS Message Server
tekserv      5009/tcp     TEKSERV    # AIPS TekServer
aipsmt0      5010/tcp     AIPSMT0    # AIPS remote FITS disk access
aipsmt1      5011/tcp     AIPSMT1    # AIPS remote tape 1
aipsmt2      5012/tcp     AIPSMT2    # AIPS remote tape 2
aipsmt3      5013/tcp     AIPSMT3
aipsmt4      5014/tcp     AIPSMT4
aipsmt5      5015/tcp     AIPSMT5
aipsmt6      5016/tcp     AIPSMT6
aipsmt7      5017/tcp     AIPSMT7
```

The `ssslock` service was introduced in `15APR98`; you will need to add this if you are upgrading from an older version of *AIPS*. See the `INSTALL.TEX` file in `/aips` on our ftp server if you want to cut/paste it, about 83% of the way down the file around line 3450. NOTE: the `INSTALL.TEX` file in `$DOCTXT` is for the version of *AIPS prior* to what you are installing.

This will enable the TV, message server, tek server, and up to 8 `TPMON` dæmons (see below) to work. If you already have these services installed, no additional work is necessary. (Reminder: the `TPMON` dæmons are to allow remote systems to access local tape drives or disks).

If your network distributes these services via NIS or NIS+ (what used to be called "yellow pages"), you may have to ensure that the primary service names are in lowercase, as shown above. SunOS 5 based NIS servers are known to have problems with service names in uppercase, hence the above names are in lowercase.

For those of you with standalone workstations or PC's, you will need to set up a dummy network interface. This is to enable the system to access the `SSSIN` and other services when you are operating "standalone". It can be achieved by adding these lines to your /etc/rc.d/rc.local or equivalent file:

```
ifconfig dummy myhostname
route add myhostname
```

* Some sites may have this file distributed via a NIS/YP map; check via the "ypcat services" command, or ask your systems administrator

You can also have your systems administrator enter these commands (as root) interactively to make them take effect immediately. The rc.local file should be one called in the boot process.

7.6 AIPS.BOOT and TPMON

You may have noticed the file AIPS.BOOT in the aips root area. This file may *optionally* be run whenever an *AIPS* host starts up. If you choose to run it (and it is *not* essential), you should use the AIPS account (not root!), or whatever account most frequently runs AIPS. So far it only has one function: to start the TPMON dæmons. This may not be necessary, as AIPS will automatically start these up if needed on your local machine everytime an *AIPS* session is started. Use of the tp= command line option can also start up the TPMON dæmons on a remote system.

The TPMON dæmons are the tasks that allow the remote tape feature of *AIPS* to work. All AIPS.BOOT does is to call the STARTPMON script in $SYSUNIX which will start up $n + 1$ servers on a given host, where n is the number of tape drives. Each TPMON dæmon handles remote connections for a given tape drive, and the extra one (TPMON1) is for FITS disk access.

If you plan on doing this — and it is *optional* — you may choose to make the $LOAD/TPMON* files set-uid to the aips account if you have one. This is done via:

```
chown aips $LOAD/TPMON*          (if necessary)
chmod u+s $LOAD/TPMON*
```

Then you should try to have your system administrator insert a line in the rc.local or similar startup script like this (from a SunOS 4 system):

```
#
# Start AIPS things
#
if [ -f /home/aips/AIPS.BOOT ] ; then
  echo "Starting AIPS daemons" >/dev/console
  su aips </home/aips/AIPS.BOOT >/dev/null 2>&1 &
fi
```

The First TPMON dæmon is for remote access to your $FITS area; if you do not wish to allow this, you should not have AIPS.BOOT used, and should terminate the TPMON1 process anytime it starts up. All TPMON dæmons can be inhibited by using the tpok option in the AIPS startup.

7.7 Defining the AIPS command

The INSTEP1 process should have created two symbolic links called AIPS and aips in the $SYSLOCAL directory (you will have probably seen this after the section about $AIPS_ROOT). If you use the C shell, it may be necessary to rehash before the command can be used (for the BASH shell, use hash -r), otherwise no action should be necessary; typing aips or AIPS should work.

There is an *AIPS* manual page in $SYSUNIX/manl called AIPS.L that you can either copy to any local manual page area that your system or site may have (*e.g.*, in the directory /usr/local/man/man1), or just use as is. The AIPSPATH.SH and .CSH scripts, called in turn from LOGIN.SH or .CSH, will extend the MANPATH variable to include this area. The readline manual page is also there, as is a symlink so that man aips and man AIPS both work.

7.8 Running the Dirty Dozen Tests

If you wish to run the DDT tests [at all], they should be run at this point in the installation (refer to "AIPS DDT History", Section 3: "How to Run the DDT" [*AIPS* memo 73; see also memo 85 for info on how the DDT has changed]). However, unless you are definitely porting *AIPS* to a new architecture, this step is unnecessary, especially for the usual systems (Sun4, Sol, Sul, Linux, AxLinux, Alpha, HP, SGI, IBM). On the other hand, it doesn't take too long and will give you peace of mind that *AIPS* is working correctly. If you had problems with insteps 2 or 3, you may decide it's best to run the test. The AIPSmark(93) benchmark value is derived from the time taken to run the Large version of the DDT; see memo 85 for details.

If you have the *AIPS* CD-ROM from NRAO, you should have the DDT images on it. Likewise if you ordered a DDT tape from us. Otherwise, you can use anonymous ftp to aips.nrao.edu, move to directory /aips/FITS, and you will see all the main DDT data files there in FITS format. The small DDT files start with DDTS, the medium with DDTM and the large with DDTL; *e.g.*, DDTSMXMAP is one of the small images (an image made with MX). Be careful before you decide to pull over everything; the large dataset totals over 50 Megabytes. Our System administrators, and your Network Administrators will thank you if you only get the large ones at off-hours!

The DDT data were recomputed for the first time in many years with the 15JAN94 release; hence the name on the directory containig the images. Also, the procedures have been somewhat revised and the DDTEXEC procedure can now read FITS disk files directly, as long as you preserve the names as found on aips.nrao.edu. Just define an (uppercase) environment variable that points at the location where you put the FITS files (no need if they are in the $AIPS_ROOT/FITS area; this is already referenced by variable FITS) and set the new DDTDISK adverb accordingly, *e.g.*, setenv RFITS $HOME/DDT and then after starting up *AIPS*, DDTDISK = 'RFITS'.

Instructions for running the DDT test can also be obtained online within AIPS via the command >EXPLAIN DDT; A short printout will describe the test and the interpretation of the results). This step takes about 5-10 minutes for the small test on most smaller Sparc systems.

For those of you who prefer to plunge ahead without reading the fine manual, here is a quick and dirty recipe. Place your DDT files in the $AIPS_ROOT/FITS area. Now start up AIPS using the NOTV and TPOK options, then RUN DDTLOAD once you're in. This should generate close to 1000 lines of output. Now type the following:

```
> tcode 'READ'
> tmode 'M'
> ddtsize 'LARGE'      (or 'MEDIUM' or 'SMALL')
> edgskp 12            (8 for medium, 4 for small)
> terse 0              (needed if you use LARGE and want AIPSmarks)
> ddtdisk 'FITS'
> tput ddt             (DON'T FORGET THIS!!!!)
> clrmsg
> run ddtexec
```

This will load the images to disk. If you want to generate an AIPSmark(93), make sure you use the `'LARGE'` DDT size. Once these are loaded (`CLRN; CATALOG` will show 11 entries), set these additional parameters:

```
> tcode 'TEST'
> outprint 'HOME:SOME.FILE.NAME'
> tput ddt
> clrmsg
> run ddtexec
```

For AIPSMarks, you want to do this on an unloaded system, probably with X11 running and a minimal compliment of tools (a single xterm, a console, a load meter perpaps) and no-one else on the system. On some Suns, the raw console (white screen, black text) is a lot slower than what you get in an xterm, and may slow the benchmark down. It is recommended that you do NOT use a message server (if one appears, kill it by typing control-c in it). To interpret the results, read *AIPS* memo 85 (accessible via our web pages and anonymous ftp server). The quick-and-dirty recipe is to figure the elapsed time (from the `PRTMSG` printout in the output file specified by `OUTPRINT`) between the two messages `RUN DDTEXEC` and `PRINTING ANSWERS, ERRORS, OTHER IMPORTANT MESSAGES` in seconds, and divide 4000 by this (so if it takes 500 seconds, that's an AIPSMark of 8).

Remember to have `SETPAR` parameter 35 set to your best guess at the *AIPSmark* before you start; you may in fact want to iterate, modifying the value and then re-running the DDT to get a better value. In reporting any result to NRAO, or quoting it to others, please make sure you specify what this parameter was as well as the actual *AIPSmark*.

8. INSTEP4

This step should be skipped if you are doing a binary or CD-ROM installation.

INSTEP4 requires a long time to compile and link all *AIPS* tasks but usually runs without problems — unless you run out of disk space. Run it in the same manner as INSTEP3. If a task fails to compile correctly it can be skipped by inserting a "-" before the task name in one of the `*.LIS` files in the `$AIPS_VERSION/$ARCH/INSTALL` directory. INSTEP4 uses the files `AIPPGM.LIS`, `AIPGUNIX.LIS`, `APLPGM.LIS`, `APGNOT.LIS`, `APGOOP.LIS`, `QPGOOP.LIS`, `QPGM.LIS`, `QPGNOT.LIS`, `QYPGM.LIS`, `QYPGNOT.LIS`, `YPGM.LIS`, and `YPGNOT.LIS`. If you know how to fix the sick task, re-compile it with the script `COMLNK`, *e.g.*:

```
$ COMLNK $QYGPNOT/SICKTASK DEBUG↓
```

9. UPDAT — CONVERTING OLD USER DATA

The 15JAN95 version was the first in several years to include a format change. This means the user data on disk for older versions cannot be accessed by 15JAN95 or later versions, and conversely, data written from 15JAN95 or more recent versions cannot be accessed by the older versions. They can, however, coexist in the same data areas without causing serious problems (other than disk space!)

There is a program called UPDAT that can be run from outside of AIPS, which will allow the *AIPS* manager to convert user data. It permits a range of usernumbers and a range of disk numbers, and as it is initiated via the RUN shell script, you can use the DA= option to select exactly which disk(s) you want defined when you run the program. If you do *not* have any *AIPS* data from a version prior to 15JAN95, then you can safely skip this whole section. Otherwise, read on and be careful.

9.1 Preliminaries

This format change was a fairly simple one. It was made to enable more than 255 extension files to be associated with a given catalog entry. A given format in *AIPS* is indicated by the third letter of most of the system files, so for older versions this was a "C" and now it is a "D" (*e.g.*, CAC00100.001;1 was the catalog entry for user 1, now it will be CAD001000.001;). We have also changed the way numeric quantities are encoded into the filenames. Instead of using hexadecimal or base 16, we now use base 36, what we call "extended hex". This has increased the number of several things possible in AIPS from 15 to 35, and in some cases from 255 to over 45,000.

Before you start updating any data, you **MUST** check for the existence of "D" format data in all disks that you are converting. For example, suppose you intend on converting all users in one data area, you should "cd" to that area first and give the command `ls ??D*` to see if there are any "D" files already there. Because the UPDAT program works by a renaming operation, it will fail if some files it wants to create already exist. If this occurs, you will be left with a real mess and it will take quite a bit of effort (and detailed knowledge of AIPS file-name mapping) to unscramble it.

If you wish to check for the existence of "D" format data for a specific user, use the $SYSUNIX shell script EHEX to find out that person's user number in extended "hex". For example, the author's number is 370, and typing `EHEX 370` will print out AA. Therefore, all new "D" format files for user 370 in a given area can be listed via the command `ls ??D*.0AA*` (from Unix). A given user's old "C" format data can be listed also, by using the Unix dc calculator program: `echo "16 o 370 p q" | dc` will print 172, which is 370 in hex; and then the command `ls ??C*.172*` will show the old files.

Most likely all you will find for new files already existing are just empty catalog files (CA* files with no CB* files at all) and trivial save/get files (SG*); these can be summarily deleted before you run UPDAT. If there is more present, you should contact the user.

9.2 Running the UPDAT Program

The program is started from the Unix command line with `RUN UPDAT` (or `RUN UPDAT DA=FOOBAR` to choose disks on system FOOBAR). It will ask you to choose from a range of users or having the usernumbers in a text file; choose the former as the latter is not very easy to use. Then it asks for the user number range. For all users, enter `1 4095`, or for a single user (*e.g.*, 370) enter `370 370`. The next question it asks is what disk number range to use. You need to enter something here despite the "default" message. Finally it asks the oldest version data; use the default here (i.e. just press RETURN).

The program then presents you with the information you entered, and gives you one last chance to re-enter it. Once you tell it to proceed, it will do the conversion. You should keep a log of all conversion you do just in case there are problems (cut/paste the output and save it in a file if necessary).

10. CLEANUP

To conserve disk space, you may want to remove some files such as these:

```
$ rm -f $AIPS_VERSION/$ARCH/INSTALL/*.LIS↓
$ rm -f $AIPS_VERSION/$ARCH/INSTALL/*.LOG↓
$ rm -f $AIPS_VERSION/*/PREP/*↓
```

(if you intend on debugging programs in *AIPS*, you may want to preserve the contents of the PREP directory so that source code debuggers can find the preprocessed files). In addition, you can remove the parts of *AIPS* that are for architectures you don't have and don't plan on having ($AIPS_VERSION/$ARCH where $ARCH expands to SUN3, SUN4, SOL, SUL, DEC, ALPHA, AXLINUX, IBM, IBM3090, LINUX, CVEX, ALLN, HP, SGI, or CRI; leave the ones you have installed!). Any VMS-specific directory is certainly not needed for a Unix installation. If you really need to, you can remove source code as well. **DO NOT REMOVE the $AIPS_ROOT/15OCT99/HELP area under ANY circumstances!** It is needed to run AIPS. If you are still tight on disk space (and especially if you compiled the programs with DEBUG set to TRUE), you can run the Unix program strip on all the executables (*e.g.*, strip $LOAD/*.EXE). The binaries as shipped on tapes and CD-ROM (and made available via anonymous ftp) are already stripped, and for most architectures the STRIP=TRUE statement in the $SYSLOCAL/LDOPTS.SH file forces binaries to be built pre-stripped.

For binary installations, you can also remove the contents of the directories $AIPS_ROOT/BIN/$ARCH and $AIPS_ROOT/DA00/$ARCH once the installation is complete. For all installations, please do *not* delete the INSTEP1.STARTED file in the $AIPS_ROOT area as it can be used should you decide to upgrade to a future version of *AIPS*. Also, please preserve the $AIPS_ROOT/REGISTER.INFO file as it will save you some typing if you register the next version of *AIPS*.

That's it! You have finished the *AIPS* installation!! Treat yourself to a Banana Split. If you have any comments on this document (especially suggestions for improvement), please send them to the address on the front of this document. All comments, compliments, brownies, *etc.*, are welcome :-)

11. COPYING

(This is merely a copy of the Free Software Foundation's General Public License)
Version 2, June 1991

11.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

11.2 GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11.

```
BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR
THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM ``AS IS'' WITHOUT WARRANTY OF ANY KIND, EITHER
EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE
ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH
YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL
NECESSARY SERVICING, REPAIR OR CORRECTION.
```

12.

```
IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR
DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL
DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM
(INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED
INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF
THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR
OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
```

END OF TERMS AND CONDITIONS

11.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
   Copyright (C) 19yy  <name of author>

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomov version 69, Copyright (C) 19yy name of author
Gnomov comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomov' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.