AIPS++ Beta Release 0.9: release notes

AIPS++ Project
National Radio Astronomy Observatory
P.O. Box O, Socorro, NM, 87801, USA

Text last updated: 1997 September 03
HTML version available at:
`http://aips2.nrao.edu/aips++/docs/project/releasenotes/releasenotes.html`

Contents

1 Purpose

This document is a guide for beta testers of AIPS++ release 0.9. It is intended to describe the installation procedures, outline the current capabilities of AIPS++, and to describe what feedback we would like from beta testers.

The content of this document is close to that of the release notes for AIPS++ release 0.8. The main changes are to describe the new functionality.

2 Planned releases of AIPS++

It is important to understand the role of this beta release in the development of AIPS++. The AIPS++ release process has been described in a document on Planning for Releases of AIPS++. AIPS++ is continuing development at the various AIPS++ consortium sites, and rapidly evolving versions of the AIPS++ system are to be found at the consortium sites. A release is a snapshot of the development system that is suitable for export to other sites. We plan to make two major releases, one in 1997 and one in 1998. The first is designated as a "Limited Public Release" since it will contain only a subset of the functionality expected to be present eventually. The second release, in 1998, is expected to have a more rounded set of capabilities. Both of these will be preceded by explicit "beta" releases. The release described here is is the second beta release: AIPS++ Version 0.9. The expected set of releases are thus:

AIPS++ V0.8: beta release (issued February 26 1997)

- Targeted to consortium sites and a few friendly astronomers
- Contains basic environment, tools, synthesis applications.

AIPS++ V0.9: beta release

- Rectify problems from V0.8
- Add spectral line functionality to synthesis processing
- Other tools

AIPS++ V0.95: beta release

- Rectify problems from V0.9
- Preview GUI for feedback before release

AIPS++ V1.0: Limited Public Release Due late 1997

- Open release, targeted to astronomers and perhaps programmers
- Contains basic environment, tools, synthesis applications.
- GUI frontends to major Distributed Objects and applications
- Interfaces should be fixed at this point,
- Will provide environment for code development.

AIPS++ V2.0 Due late 1998

- Full synthesis package (including some functionality for VLBI)
- Single dish package
- Image Analysis package
- General toolkit
- Visualization toolkit and applications
- Contributed code

We use the term beta release to denote a distribution of the system to a limited number of people who have agreed to test the system and report problems and general comments. The principal goals of the beta releases and the LPR are to provide astronomers with a first look at functionality in a number of different areas: synthesis, image analysis, general tool-kit, and for the AIPS++ Project to get experience with distribution and support of AIPS++ outside consortium sites.

More specific description of what we expect to learn from beta-testers is given below. First, we describe the current capabilities of AIPS++ and how to install it.

3 Current capabilities of AIPS++

In this section, we provide a brief overview of the capabilities of AIPS++, describing both what is present in the system, and what we have not yet delivered.

3.1 What is present

- All user capabilities of AIPS++ are documented via the AIPS++ User Reference Manual.
- The heart of AIPS++ is a powerful command line interpreter called Glish (see Appendix A for copyright information). Glish has both a tutorial and a reference manual.
- AIPS++ has a number of general purpose tools that are accessible from Glish.
  - Access to all data in AIPS++ via the `table` module.
  - Catalogs of directories, interpreting the contents to show different types of files, and catalogs of potentially everything in Glish (*i.e.* functions and variables).
  - Plotting of Glish variables.
  - Assorted mathematical capabilities such as statistics, random numbers, polynomial fitting, interpolation, Fast Fourier Transforms and convolutions.
  - Manipulation of measured quantities with units and reference frames either from the Glish command line or via a gui.
  - Display of AIPS++ or FITS images or Glish arrays using the aipsview program.
  - Conversion of images to and from FITS image format, reading and writing AIPS++ images to and from Glish. Statistics, histograms and moments of images may be calculated. Subimaging and padding are both allowed.
  - Logging of messages to a gui window and a table, also printing to a postscript printer or to ghostview.

  - Miscellaneous useful utilities including: obtaining AIPS++ configuration information, the `help` utility, a printer tool (including a gui), execution of shell commands, reading and writing to external files using C-like commands, *etc.*

- The synthesis capabilities of AIPS++ are as follows:
  - Filling from and writing to a UVFITS file,
  - Filling from WSRT format,
  - Full imaging, deconvolution, and self-calibration,
  - Joint deconvolution of Stokes IQUV,
  - Robust, uniform and natural weighting,
  - Flexible windowing in the deconvolution,
  - Non-Negative Least Squares Deconvolution,
  - Flexible construction of models for self-calibration,
  - A sophisticated multi-component model for gain effects,
  - Interactive editing,
  - Flagging of gain solutions by antenna, spectral window, and time interval.
  - Writing of final images to FITS files.
  - Spectral line synthesis processing

  More details of the synthesis capabilities can be found from the documentation on the imager, and from the Guide to Synthesis Processing in AIPS++.

3.2 What isn't present

- GUI interfaces. These are under development.
- Fully-fledged synthesis *cross*-calibration is deferred until the major release (V1.0) in 1998. Self-calibration is present in AIPS++ V0.8 and V0.9.
- Efficient processing of large synthesis images. Current processing is optimized towards spectral-line observations.
- Single dish processing.
- A cookbook.
- We do not expect people to program at the C++ level using the beta release. If you really want to do this, please contact us separately. It can be arranged but is not supported as part of the beta release.

4 Installation of AIPS++

We assume that you are installing a Beta release of the system, complete with prebuilt binaries. You need to refer to other documentation if you are attempting an installation from source; such is beyond the scope of this document.

Installation over the top of a existing version is possible but you may wish to make a new installation.

4.1 Before installation

If you have an existing installation, we recommend that you write existing MeasurementSets and Images to FITS format from where they can be reloaded.

4.2 What will the installation do?

The installation process involves you downloading several compressed (GNU-zipped) tar files from the `aips2.nrao.edu` anonymous ftp server, and running an install script (called `configure`). It will uncompress and unpack these tar files in the right places, and you will end up with a complete source-and-binary AIPS++ system, ready to run. All that will be required is the invocation of an initialization script (`aipsinit.sh` or `aipsinit.csh`) which you can call for convenience in your `.login` or `.profile` or equivalent login script.

4.3 Requirements

This beta release is targeted at a few architectures:

`sun4sol_gnu` Sparc systems running SunOS 5

`linux_gnu` Intel Pentium or Pentium Pro machines running Linux

We use the GNU C++ compiler to generate the binaries. You do not need to have the GNU compiler, however, as prebuilt binaries and libraries are provided.

While a separate account for AIPS++ is recommended, it is not required for the installation. Your own personal account — assuming sufficient disk space — should suffice. You will not need `root` access, and should *not* perform the installation as `root`. In addition, you will not be required to gain access to any operating system files or directories; everything can be done from a normal, non-privileged account.

AIPS++ uses the Caltech PGPLOT package (version 5.1.0 or later) to provide various plotting capabilities. If you already have an installation of PGPLOT, then you can and should use that. Otherwise, for convenience we provide a copy of the PGPLOT library and the required font file, grfont.dat. Please note that PGPLOT is not public-domain software. However, it is freely available for non-commercial use. It is copyrighted by California Institute of Technology.

Here is an approximate outline of the various hardware and software **requirements** for this release:

- Sparcstation IPX-class or better, preferably at least Sparc-20, or a Pentium class machine.
- SunOS 5, ideally SunOS 5.5 (a.k.a. Solaris 2.5) or higher; or Linux 2.0 or 2.1. We expect distributions for HPUX, SGI and Dec Alpha (OSF1) to follow).
- 64 Megabytes of memory (32 is workable but your system will be swapping most of the time)
- At least 250 Megabytes of disk space (not counting space for data)

- At least 100 Megabytes of swap space

While this is a minimum list, you may also want to include these optional items:

- The PGPLOT library and font file (`grfont.dat`). These are supplied in the binary distribution, but if you already have version 5.1.0 or better, you should make sure your `PGPLOT_FONT` and/or `LD_LIBRARY_PATH` environment variables are set up for it.
- The `libtcl7.6.so` shared library. A copy is provided in the binary distribution.
- A copy of `gunzip`. The `configure` script will offer to download and install a version for you if you do not already have it (or you can retrieve the version available in the Classic AIPS anonymous ftp directories).

4.4 Fetching the stuff

1. You will want to create an "aips root" directory for AIPS++, much as many of you have done in the past for Classic AIPS. Create a `code` subdirectory underneath this AIPS++ root area, and `cd` to this subdirectory.
2. Retrieve all source savesets via anonymous ftp from the server `aips2.nrao.edu` in directory `/pub/RELEASED/aips++0.9/`. These should include at a minimum:
   (a) `README` (read it for last-minute changes)
   (b) `VERSION` (the version of AIPS++ you're getting)
   (c) `configure` (the installation script)
   (d) `aips-09.001.tar.gz` (The core of AIPS++, source code)
   (e) `doc-09.001.tar.gz` (Documentation source)
   (f) `docs-09.001.tar.gz` (Documentation)
   (g) `install-09.001.tar.gz` (Installation infrastructure)
   (h) `nfra-09.001.tar.gz` (NFRA specific code)
   (i) `trial-09.001.tar.gz` (trial, where most of the interesting stuff is to be found)

   and make sure you leave them in your `code/` subdirectory.
3. Retrieve the binary archive from the same ftp server in the same directory. This will be either:
   (a) `bin-sun4sol_gnu-09.001.tar.gz` for Solaris systems, or
   (b) `bin-linux_gnu-09.001.tar.gz` for Linux/Intel systems.

   Place it in your `code` subdirectory also. If you are using some other architecture, replace the architecture name (*e.g.*, `sun4sol_gnu`) in the above filename with the relevant architecture (*e.g.*, get `bin-alpha_gnu-09.001.tar.gz`). We will have `sun4sol` and `linux` available at the release, and others shortly thereafter; these may include `DEC`, `HP` and `SGI` depending on demand.
4. Make sure the configure script is executable (`chmod +x configure`).

4.5 Running the configure script

4.5.1 Preliminaries

Before you plunge in and start the installation, you may want to make some preparations. First, if you have a local copy of `gunzip`, make sure your `PATH` environment variable includes the directory where it is found. If it does not by default, you can add it like this (if your shell is one of `sh`, `bash`, `ksh`, `zsh` or some other bourne-like shell):

```
PATH=${PATH}:/wherever/bin; export PATH
```

or if your shell is `csh`, `tcsh` or another C-like shell:

```
setenv PATH ${PATH}:/wherever/bin
```

In the above examples, it is presumed that `gunzip` is found in the directory `/wherever/bin/gunzip`.

Another variable you may wish to add to the `PATH` variable is the location of your X11 or Open-Windows software. On Sun systems, this will typically be in `/usr/openwin`:

```
OPENWINHOME=/usr/openwin; export OPENWINHOME
PATH=${PATH}:$OPENWINHOME/bin
```

or for C-shells:

```
setenv OPENWINHOME /usr/openwin
setenv PATH ${PATH}:$OPENWINHOME/bin
```

If you have an existing Classic AIPS installation, you may want to ensure that you have defined its environment via the `LOGIN.SH` or `LOGIN.CSH` scripts first. This will cause the `configure` script for AIPS++ to automatically generate a hosts database from the existing AIPS information in `HOSTS.LIST`, which can be a real timesaver.

Finally, if you have a favourite text editor (emacs, vi, textedit, elvis, *etc.*), you should make sure that either the `EDITOR` or `VISUAL` environment variables are set to invoke it, *e.g.*,

```
VISUAL=/usr/local/bin/emacs; export VISUAL
```

or:

```
setenv VISUAL /usr/local/bin/emacs
```

as appropriate for your shell.

4.5.2 Running the configure script

Make sure you are in the `code` subdirectory where you left the GNU-zipped tar files, and enter this on your Unix command line:

```
./configure
```

Then answer the questions, which are fairly routine. The steps you have to go through for this installation are actually fewer and less complex than the equivalent (`INSTEP1`) for Classic AIPS, so don't panic.

The first thing you'll see is something like this:

```
The 'configure' script bootstraps the AIPS++ system to a state
where it can be used by end-users.  It will by default perform
a binary installation, but it can be used to recursively build
the libraries, executables, and other parts of the system.

It may be used to configure AIPS++ from scratch, or to add a
new architecture.  In the latter case the installation of the
aipsinit scripts and databases can be skipped.

You will be asked a few simple questions, most of which have
default answers (in square brackets).  Simply type <return> to
accept these defaults.
```

If you are reading this document, then you are doing a binary installation, you are not configuring from scratch, and you are not adding a new architecture. If you have previously defined an `EDITOR` or `VISUAL` environment variable, you will see a notice to the effect that `configure` found this, and will use it to let you edit a few critical files along the way.

1. **AIPSROOT**: This is shorthand for the top-level AIPS++ directory (the one in which you created the `code` subdirectory); the script will ask you to verify its correctness. If for some reason the `configure` script guesses it incorrectly, please enter the correct full directory name here. If you do not run the script from the `$AIPSROOT/code` directory, you will be prompted to move there and restart the script.
2. **Optional Utilities**: For a full-blown AIPS++ consortium installation, there are a certain number of utilities needed to make it fully functional; these will be listed now, and instructions will be presented in case you want to install any of them later. They are *not required* for the binary installation, unless you plan on doing code development within the AIPS++ system.
3. **Required Utilities**: For the from-source installation, some third party utilities are required. Only one of these is needed for the binary installation, namely `gzip` (techincally, only `gunzip` is needed but both usually come as a package). If `configure` finds `gzip` in your `PATH`, it assumes all is well and skips on to the next section. Otherwise it offers to download, build and install the utility for you. Note that root or superuser access is not required, as it will place the `gzip` *etc.* binaries it builds into a relevant AIPS++ directory.

4. **Extract Compressed tar files**: As requested by various potential users and other interested parties, the beta release for AIPS++ is including sources as well as the binaries. The existence of the source files is not required to be able to run the system. However, some source level packages — such as the install `tar.gz` saveset — are required. In any event, `configure` will unpack each GNU-zipped tar file you have placed in the `code` subdirectory automatically. It has a manual override so that you can, for example, choose not to delete the tar files once they are unpacked.

5. **Install Binaries**: This step is essentially identical to the previous one, except that the binaries and libraries needed to run AIPS++ are unpacked for each architecture you have selected. For the early beta releases, the most tested architectures are `sun4sol_gnu` and `linux_gnu` (these correspond to Sparc systems running SunOS 5/Solaris 2 with the GNU `g++` compiler, and Linux on Intel platforms also with the GNU compiler; though you do not need the compiler to run the AIPS++ binaries). However other architecures will be available at or near release (see the previous section on "Fetching the stuff").

6. **Install aipsinit**: In every AIPS++ installation, there is a shell script that one "sources" or invokes to define the environment necessary to run the system. The `configure` script automatically sets these files (`aipsinit.csh` and `aipsinit.sh`) up for you.

7. **Construct aipshosts**: The AIPS++ system is flexible enough to permit you to alter its setup in a per-host, per-architecture, and per-site manner. In order to do this, it needs to know the name and architecture of each computer in your AIPS++ site. To achieve this, a text file called `aipshosts` will be created in the `AIPSROOT` directory. This is the exact equivalent of the Classic AIPS `HOSTS.LIST` file, and in fact the `configure` script will attempt to use the information in the latter if it is found. If you elect not to use this info (or if it's not present or found), you will be asked for a site name, the architecture, and a description of the current host and each additional host you want to add to the AIPS++ site.

8. **Fetch patches/bug fixes**: *If you are installing for the first time you can ignore this item.*

   If you already have a binary installation and you wish to install a new patch you should merely need to fetch the latest bin- and docs- (note! plural) .tar.gz files and untar them from the `code` directory. Be sure to temporarily undo any "noclobber" setting in your shell to allow replacement of files by the (un)tar command. The source files will be out of date, however for a binary installation this should be fine. If you want the sources to be current as well, it is probably best for you to reinstall from scratch.

9. **Construct aips++local file**: The purpose of this file is as an aid to the `aips++` command, so that local definitions of, *e.g.*, `PGPLOT_FONT` and `LD_LIBRARY_PATH` may be made to accomodate the PGPLOT and other libraries needed to run the AIPS++ binaries. The `configure` script will check for an existing `PGPLOT_FONT` definition, and in its absence set it to point at the AIPS++ -provided location for the PGPLOT `grfont.dat` font file. In addition, on Sun and Linux architectures, a definition or revision of the variable `LD_LIBRARY_PATH` will be made so that the `pgplot` and `tcl` shared libraries may be used (these are provided with the binary installation, though if you have local copies they will be used first). You may have to edit this file to set or correct the value of this latter variable; in particular, you should make sure it includes the location of your X libraries (*e.g.*, `/usr/openwin/lib` or `/usr/local/X11/lib`) and the library location of any local installation of PGPLOT. Note that the variable should be a colon-delimited list of directories.

10. **Construct aipsrc and makedefs**: For more generic parameters, these files are used. The former generally specifies quantities such as the AIPS++ manager account name (this need not be a dedicated account) and the default printer and paper format (A4 or Letter), while the

latter is generally only of interest if you intend on code development or building the system from source. The `configure` script will ask you to edit the site-specific `aipsrc` file and gives instructions on what to change; there should not be much you need to edit (just check what `configure` did). Hopefully the comments are clear enough to avoid any confusion. The file must be edited, as `configure` checks if the word `DEFINE_ME` has been removed from it before proceeding.

At this point, you should have an AIPS++ installation that is ready for testing.

4.5.3 Testing the installation

To test the installation, use the assay module:

```
.      /aips++/beta/aipsinit.sh       # For Bourne/Korn/Bash shells
source /aips++/beta/aipsinit.csh      # For C-like shells
aips++
- include 'assay.g'
- assay().trytests()
```

This will run all registered test scripts in the system, and summarize at the end which failed. The complete sequence should take about 20 minutes.

The final number printed will be the number that failed. If this is zero, then the installation has passed the first test. If it is not, then you need to determine which test failed and run it again *e.g.* if `imagertest` failed then do:

```
aips++
- include 'assay.g'
- assay().try('imagertest')
```

If it is not clear why a particular test fails, then please contact us using the mail exploder aips2-beta@nrao.edu (see below for more details).

If no tests failed then you are ready to move on to using the system. At this point, it may be a good idea to re-read the document on Getting Started in AIPS++, and decide what you wish to try out.

4.6 Things That Can Go Wrong

4.6.1 ld.so (shared library loader) problems

The most likely source of error on systems that support it is an incorrect value for the environment variable `LD_LIBRARY_PATH`. On Sun and Linux systems, this variable specifies additional directories to search for unresolved shared libraries referenced in the binaries. Your site may not have the various system and optional shared libraries in the same places as we do, so the binaries we generate may need this variable set to include some of your site-specific directories before you can run certain programs, or the aips++ system itself.

The `configure` script will attempt to figure out what is needed for this variable, and it will set and/or extend it (possibly on multiple lines) for you in the file `sun4sol_gnu/aips++local.sh` (replace `sun4sol_gnu` with your architecture if needed); while it tries to figure out the right settings, it may not be correct for your environment.

If you find the value in this file incorrect, please change it to whatever is correct for your system. Note that if you do not have either PGPLOT or TCL libraries on your system, this variable will need to include the AIPS++ architecture-specific library (*e.g.*, `$AIPSROOT/sun4sol_gnu/lib`) as that is where they will then be found.

5 Feedback requested from beta testers

The purpose of the beta release is two-fold: to determine and correct bugs, problems and deficiencies in the current software, and to solicit general comment and feedback on AIPS++. To be successful, both require a considerable allocation of time and patience, and so we would like to thank you for agreeing to be a tester.

AIPS++ is a totally new package and has many features and capabilities that most astronomers will be unfamiliar with. It has been designed in accord with the AIPS++ User Requirements to have both canned algorithms and general purpose tools. The general flavor of the package may may seem strange (and probably irritating!) at first. Please give it a chance. What you see here is undoubtedly rawer than what the end-user will eventually encounter. We do plan to add more user-friendly interfaces as we gain experience in what people want.

We suggest that you first take time to read a number of key pieces of documentation. Please remember to keep notes on the documentation that you read: Is it clear? Does it cover what you need to know? Is there jargon present? Do the examples actually work?

Start by reading the document on *Getting Started in AIPS++*, probably all of the easy track. Then you should be in a good position to try out some of the examples in that document.

If that leaves you curious about Glish, then try the *Glish tutorial*. This document contains a very good introduction to the use of Glish, and also touchs on some more sophisticated uses of Glish.

If you are interested in synthesis processing, then we recommend that you read the documentation on the imager and the *Guide to Synthesis Processing in AIPS++*. imager has a demonstration (imagerdemo) and a test (imagertest) that are both worth running. After running these examples, you might want to read the introductory material for imager again.

Another tool that is worth looking at in some detail is Wim Brouw's measures module. This handles conversions of measured quantities with units and reference frames. For example, it can be used to calculate the rising time of a source at a given observatory.

At this point, you should be ready to decide on a small project that you might wish to pursue, perhaps synthesis imaging of some sort, or processing of ascii table data using the table system, or simply image display.

We suggest that as you proceed, you report bugs and "features" as you find them using one of the various methods described in the next section. We also need more general and wide-ranging reactions to the software, but our experience with our testers so far is that you should give the system a little time beforehand.

6 How to get help on using AIPS++ and how to report bugs

If you are having trouble doing something that you think the system should do, and you've read the documentation, you can get a human to help by sending a message to the AIPS++ Beta exploder `aips2-beta@nrao.edu`. This mailing list will include all the beta testers, and members of the AIPS++ Project. You should expect to receive advice from a member of the Project within a day or less. If you see a message from one of the other testers that you understand the answer to, please help us out and reply. An archive of all messages is available.

If you know what to do and it just doesn't work, then report a bug. There are a number of ways of doing this. The easiest is to use the `bug()` command from inside AIPS++. The advantage of this approach is that it gives us information we need about your environment. An alternative and sometimes more straightforward approach is to use the WWW interface to the AIPS++ Bug reporting system. Please also use this mechanism to request or suggest enhancements to the current system. You can also review all existing bugs from the AIPS++ Gnats Page.

Finally, if you are totally frustrated and everything else has failed, contact Tim Cornwell (505 835 7333, tcornwel@nrao.edu) or Brian Glendenning (505 835 7347, bglenden@nrao.edu)

7 Known problems

We are aware of the following problems:

gplot1d under Linux Extra click needed to set focus. So, for example, to set the Select option, you may have to click *twice* on the Select button.

8 Changes visible to users

8.1 Changes between 0.9.01 and 0.8.02

There have been many changes:

Aipsview Many improvements to facilitate control of the program. Can also provide flexible slicing through multidimensional images. See the Aipsview Manual for more information.

Glish Now considerably more robust. It traps `control C`, and `Control \` will exit.

Gridding A serious gridding bug that could occur under some circumstances has been fixed.

Hanning smoothing Hanning smoothing and numerous small improvements have been made to the Glish/image interface.

History FITS history cards are now preserved for images.

Image Asynchronous execution is possible (as for other Distributed Objects). Production of moments images is now possible.

Image cache The cache size for images could be too low, causing unnecessary I/O.

imager Many changes involving spectral line processing. A demonstration of spectral line processing is available as `imagerspectraltest()`. There have been improvements to the speed of all types of processing, and a number of changes to the interface. Imager now traps errors more carefully, and can be interrupted (by Control C) and restarted (by `di.restart()`. Filling of MeasurementSets is now possible from BIMA and WSRT data sets.

Large Events/Alpha Glish could hang when sending very large events. Also a bug only affecting DEC Alphas has been fixed.

Log Window The log window now builds all at once rather than incrementally.

Messages Some unnecessarily obscure messages have been eliminated or made less threatening.

Spectral Cubes The FITS/spectral axis interconversion is more robust now.

8.2 Changes between 0.8.02 and 0.8.01

UVFITS Some unusual UVFITS files (SOURCE random parameter but no AIPS SU table, spaces in embedded spaces in the UU/VV/WW random parameters) can now be read.

visset.g Editing was not updating data on disk correctly, and editing was only active on the first call to edit. The editing has been reworked considerably in response to a number of bugs. The function `vissetdemo()` demonstrates the capabilities of visset.

Glish Numerous bugs in Glish have been fixed. See the bug-tracking web pages for details.

Logging Updating the log window from the log table is now much faster. Multiple SEVERE errors will now go in the same window. Some bugs have been fixed, and a global variable "t" is no longer created.

Measures A sign error in radial velocity calculations has been corrected. A change in the calculation of `B1950->App` conversions has been made (a change in the order of aberration and precession corrections, and the details of elliptical aberration corrections).

Plotting Glish `setprintcommand()` has been added to program the functionality of the gplot1d `File|Print` button. For example, `setprintercommand("ghostview")` will cause ghostview to receive the temporary output file.

Glish `setprinter()` should no longer be used. For backward compatibility, the function `setprinter(<printerName>)` has the same effect as `setprintcommand("pri -P<printerName>")`.

Glish `psprint()` now causes a dialog box from `"printer.g"` to appear to control printing for each plot.

`gplot1d` will create its own private colormap if enough colors are not available in the default colormap. This may cause flashing when moving the mouse between the plot window and other programs. If you do not want this behaviour, close the plot window, then close other applications that use excessive colors (*e.g.*, Netscape), and reopen the plot window.

All boolean functions that were previously restricted to accepting the set of strings (`"on"`, `"off"`, `"true"`, `"false"`, `"yes"`, `"no"`) now also accept boolean or numeric values. For scalar values, the value zero is false, and all other numbers are true.

All plotter functions will now fail gracefully if any parameter is of an unacceptable type.

Catalog catalog and gcatalog now use the standard bug form. catalog will not crash when "unusual" files are in the directory any more.

Printing The aipsrc variable `printer.default` is now used if no `PRINTER` environment variable is set.

table.g A bug in the creation of zero-row tables has been corrected.

DEC alpha Numerous changes to allow the beta release to compile with the latest version of Digital Unix have been made.

Bug reporting Glish will no longer crash in the bug-reporting form when control-characters are entered.

Documentation Numerous clarifications and corrections have been made to the documentation. GIF icon files for documents generated via LaTeX2html are now obtained locally rather than over the network.

Help Help will now start up faster and it is be less picky about the format of the strings it is given.

aipsinit If an architecure of type "NONE" is supplied for the host "DEFAULT" initialization will be suppressed for hosts which are not explicitly listed in the `aipshosts` file. The various aipsinit scripts will work now when there are a lot of files in the directory.

Memory use Memory used to access images will be smaller in some circumstances. However much larger decreases in memory use will appear in version 0.9.

configure The configure script should do better at finding the correct timezone offset.

8.3 Changes between 0.8.01 and 0.8.00

Glish does not start A bug where glish could not create the necessary keys has been fixed. This could make Glish unusable.

Wrong times in log messages A bug which caused some log messages to not apply the timezone offset has been fixed.

Bug reading FITS images FITS images without any sky-plane astronomical coordinates could cause bus errors in the FITS reader.

Local help files broken Local help files sometimes did not display properly in a browser.

9 Acknowledgements

We thank Tim Pearson and Martin Shepherd for allowing use of PGPLOT, and for help with the Glish binding to PGPLOT.

Appendix A: Glish Copyright

Glish is covered by the following copyright:

Appendix B: AIPS++ Copyright

AIPS++ is covered by the following copyright:

Correspondence concerning AIPS++ should be addressed as follows:

Internet email: aips2-request@nrao.edu

Postal address: AIPS++ Project Office, National Radio Astronomy Observatory, 520 Edgemont Road, Charlottesville, VA 22903-2475 USA.