2 Starting Up *AIPS*

This chapter contains general information concerning the steps needed to obtain access to, and use, an *AIPS* system. It attempts (as does the design and coding of *AIPS* itself) to avoid specific references to particular computer devices and to the peculiarities of any one *AIPS* installation. We will assume, for the most part, that you will be running *AIPS* on a Unix workstation although *AIPS* should still work in more classical environments. Even for workstations of the "same" operating system, some installation-specific practicalities remain. For the NRAO installations, these are described in Appendix Z.

2.1 Obtaining access to an *AIPS* computer

Most *AIPS* sites now possess a number of computers which are networked together and are each individually capable of running *AIPS* while sharing both disk resources. Most such computers cannot support more than a few simultaneous users (or simultaneous incarnations of the same user) of *AIPS*. Thus, most locations are obliged to institute a mechanism for distributing the available *AIPS* time to the people desiring it. At NRAO, some of the computers are assigned to individual staff members and are normally used only by them. Other computers, including all of the most powerful ones, are for "public" use, but are mostly still on an assigned basis. You should arrange to have a workstation assigned to you for your *AIPS* processing. A few of the computers are available on a first-come, first-served basis, and are often used remotely. There may be sign-up sheets and rules for their use posted in or near the principal "*AIPS* Caige" (user-terminal room). To promote fair and efficient use of the system, there are often restrictions on the amounts of time that any one user or user group may reserve.

AIPS can support several simultaneous users which it calls `AIPS1`, `AIPS2`, *etc.* In the workstation environment, this is used primarily to allow one user to have separate simultaneous *AIPS* sessions using multiple windows. This also allows users to log in to remote computers (*e.g.*, with the Unix tool `slogin`) and run *AIPS* while remaining comfortably ensconced in their offices in front of their own (presumably lesser) workstations. You should not do this, of course, without permission.

2.2 Using the workstation

The way that a workstation behaves is a function of the type of workstation, the computer operating system, the window manager program, and the set-up files for the specific computer account being used. Given all these variables, it is hard to give detailed usage instructions. Nonetheless, it is important for beginning users to master the foibles of the workstation(s) they will be using.

2.2.1 Logging in to the workstation

Find your assigned computer in the appropriate *AIPS* caige or office, or an available one intended for general use (checking any sign-up sheets for it). Typing C_R on the keyboard will reveal the current state of the workstation. If you see a message prompting you to log in (*e.g.*, `AOC RedHat Linux, [monkey] login` on a Socorro Linux workstation named monkey), then the computer is ready for you to log on. Type the account name you are supposed to use for *AIPS* followed by a C_R (use `Tab` in forms) and then type the password (it will not be visible on the screen) followed by another C_R. See your *AIPS* Manager for the account to use and its password (which should change with time). Many sites will assign an account to you personally, while some use a more generic AIPS account. The login scripts should start the window system automatically and produce one or more `xterm` or `aixterm` windows that you can use for starting `AIPS`.

If the initial C_R produces instead a set of windows (and/or icons), the computer is already being used. If these windows include the AIPS TV and possibly the `TEKSRV` and `MSGSRV` server windows, it is being used for `AIPS`. Check with other possible users before proceeding. If it's okay to use the system, you should log the previous user out and log in for yourself, restarting the window system. If you are patient, you can open each iconified window (by clicking on it once or twice), see what it's doing and finish up and/or exit. If the prompt is > in any text window, `AIPS` is running there and you should type:

> KLEENEX C_R

which will get out of `AIPS` and kill the servers. Then once at the system prompt (Unix), you can type exit C_R (lowercase!) to make the window go away. If the `XAS` AIPS TV server is still running, just press the escape key while the cursor is in the TV window. For the `MSGSRV` message server, move the cursor into the window and press `CTRL C`. Finally for the Tek server, hold the control key down while you press the left mouse button, and choose the `QUIT` option.

The procedure for exiting from the windowing system will depend on what window manager you use. If your system uses `KDE`, there will be an icon on the icon bar with a large `K` superimposed on a globe. If the system uses Gnome, then the magic icon is an image of a foot. Move the mouse to the icon and hold down the left button. A pull-down menu will appear; select the `Logout` function.

2.2.2 Control characters on the workstation

To correct characters which you have typed, you may have to press either the `BackSpace` key or the `Delete` (or `DEL`) key. Unfortunately, which is required varies with the application you are using and how the AIPS account (or your personal account) has been set up. For details, see the manual page on `stty` with particular note of the `erase` function.

A control character is produced by holding down the `CTRL` (or `Control`) key while hitting another key. Some control characters under Unix have characteristics that may confuse users more used to other environments (VMS, MS-DOS). In particular, `CTRL D`, `CTRL T`, `CTRL Y` and `CTRL Z` behave much differently under Unix than under VMS. `CTRL D` is used in Unix as a signal to logout, unless otherwise inhibited. If you use the AIPS accounts at either Charlottesville or the AOC, this feature is automatically disabled. While in `AIPS`, `CTRL D`s are interpreted from the > prompt as an `EXIT` command. `CTRL T` (under GNU `readline`) transposes two characters, while `CTRL Y` inserts characters previously saved in the "kill" buffer. `CTRL Z` suspends the current process, printing `Stopped` on your window and leaving you at the Unix prompt level. The `Stopped` message does *not* mean that the process has been terminated. It simply means the process has been suspended and placed in the background. For AIPS users, the suspended process is normally `AIPS`*n*. Users who do not understand this state often start up another AIPS session. In doing so, they are tying up a second AIPS number. If a user does this enough times, s/he can eventually tie up all available `AIPS`*n*'s. If you are unfamiliar with the use of `CTRL Z` (suspend) in Unix systems, it's best not to use them from AIPS, unless expert advice is close at hand. With an X-Window display, it is preferable to pop up a new `xterm` or other window and do whatever you want in it, leaving the AIPS session undisturbed. (You can get a new `xterm`, usually, by moving the cursor into the root (background) window, pressing the right mouse button, and selecting the appropriate option.) If you have suspended the current process (usually `AIPS`) with `CTRL Z` to get to monitor level (for instance, to edit a `RUN` file), then you can bring the suspended process back into the foreground with the command fg C_R.

To abort any execution in your window, type `CTRL C`. Using `CTRL C` while in `AIPS` will unceremoniously eject you to the Unix prompt. You will have to restart `AIPS` with all the input parameters having been lost. In some cases, any AIPS tasks running in the background, and maybe even the TV and other servers, will also be "killed" and will disappear from the screen. Aborting AIPS "tasks" (sub-processes) is usually done from within AIPS with the command ABORT *taskname* CR (see § 3.1.2) rather than with CTRL C's or Unix-level system commands. Not only does this avoid killing AIPS, but it even allows for orderly deletion of scratch files.

During execution, scrolling of output lines out of the window can be halted by typing CTRL S and resumed by typing CTRL Q. If you are using an xterm (or cmdtool or aixterm) window with a scroll bar, you probably won't have to worry too much about doing this; use the scroll bar to review lines which have rolled off the visible part of the window. You can specify how many lines these terminal emulator windows remember, *e.g.*, for xterm with the -ls option or with the X resource xterm*saveLines (in your .Xdefaults or .Xresources file).

2.2.3 Starting the AIPS program

As you enter the commands needed to log in to your system and start AIPS, please read all messages which appear. They are often important and relate to current system, disk, and *AIPS* problems which may affect your reductions.

To begin AIPS, enter

% aips CR with no options initially

You will then be shown a list of printer devices and be prompted to Enter your choice:. You will then be told about the assigned printer queue and data disks. If all is going well it will then tell you

You seem to be at a workstation called *monkey*

Starting local TV servers on *monkey*

where *monkey* is the name of your workstation. Any news messages about your *AIPS* installation will then appear. Read them; they might be important. Finally, you should see the messages:

Starting up 31DEC24 AIPS with normal priority

BEGIN THE ONE TRUE AIPS NUMBER n (release of 31DEC24) at priority 0

where 31DEC24 identifies the release of *AIPS* and n is a number between 1 and 6 (typically). If this is the only *AIPS* session on the computer, you should be assigned $n = 1$, with higher numbers used for additional sessions. If you start with $n > 1$, someone else may be using your computer remotely. AIPS will then tell which TV and graphics devices have been assigned to you:

AIPS n: You are assigned TV device *nn*

AIPS n: You are assigned graphics device *mm*

where *nn* and *mm* are numbers assigned to your workstation (or, rarely now, to real TV and graphics devices). AIPS will now ask you for your user number and provide a ? prompt:

AIPS n: Enter user ID number

? *uuuu* CR

where *uuuu* is the number assigned to you for the local *AIPS* system (in decimal form). The AIPS prompt > should now appear.

There is more. Notice the line above that says "starting local TV servers on monkey"? At that point, the process of figuring out what computer you're running on and what display you're sitting at (they may be different) is shed in an asynchronous way while the main process of starting the AIPS program proceeds. Then, sometime later, you will see the following messages appear in the same window:

```
XASERVERS: Start TV LOCK daemon TVSERV on monkey
TVSERVER: Starting AIPS TV locking, Inet domain
XASERVERS: Start XAS on monkey, DISPLAY monkey:0
XAS: ** TrueColor FOUND!!!
```

```
XAS: *** Using shared memory option for speed ***
XAS: Using screen width height 2540 1920
     max grey level 8191 in 16 grey-scale memories
     characters multiplied by 2
XASERVERS: Start graphics server TEKSRV on monkey, DISPLAY monkey:0
XASERVERS: Start message server MSGSRV on monkey, DISPLAY monkey:0
```

Each of the first four messages should announce the starting of one of the servers. The Tek server will appear in iconified form somewhere on the screen, while the message server will appear opened (not iconified) somewhere else. Finally, the `XAS` TV server appears in opened form. An environment variable can be set to have the Tek server appear in open form and a `.Xdefaults` (or `.Xresources`) file option may be set to have the TV appear in iconified form. If your X-Windows supports 24-bit TrueColor, then `XAS` will use it. Otherwise, `XAS` will use 8-bit PseudoColor which is faster but less flexible. In this case, if you have a lot of colors in your X11 display (*e.g.*, an image on the root window displayed with `xv`), you may also get the message:

```
XAS: Using screen width height 1142 800, max grey level 189
XAS: Warning -- creating virtual colormap
```

which means `XAS` wasn't able to find enough free colors in the main colormap (189 in the above example) and had to create its own. In this case, the colors of every other window will "flash" when you move the mouse cursor into the opened `XAS` TV window, and vice versa. You can use `xsetroot -solid navy` command (or other legal X colors) to blank out whatever is on the root window; then restarting `AIPS` will restart the TV server, hopefully without a virtual colormap. There are a number of X-Window parameters which may be specified in your `.Xdefaults` or `.Xresources` file for these three windows. After `AIPS` begins, type HELP XAS C_R, HELP MSGSRV C_R, and HELP TEKSRV C_R for details. Among these is a parameter controlling how many colors `XAS` tries to use in PseudoColor and whether it tries to use TrueColor or not. Character size multiplication occurs by default when you have a large number of pixels on your screen. It can be controlled by an `.Xresources` parameter and the `AIPS` verb `CHARMULT`. See § 2.3.2 for more information about `XAS`.

There are several options you can use in starting up `AIPS`. To see them, just enter `man aips` at the Unix command prompt, or if you are already in `AIPS`, type HELP AIPS C_R. This information is reproduced in part below:

```
aips [OLD, NEW, or TST]
     [TV=[disp][:][host]]
  or [TV=local[:n]]
  or [NOTV]
     [TVOK]
     [DA=host[,host,...]]
  or [DA=default]
  or [DA=all]
     [PR=#]
     [REMOTE or REM or TEK]
     [DEBUG[=prog][:aips]]
     [LOCAL] [NORL] [NOEX]

DESCRIPTION
       The  sips  command  starts up the AIPS command interpreter and associated AIPS
       server processes.

OPTIONS
       All command line options are case insensitive.

       AIPS allows up to three versions to co-exist (disk space  permitting)  in  one
```

```
installation.  They are identified either by date (e.g. 15OCT98) or name (OLD,
NEW, or TST).  On most installations, these will all be the same.

old     Start the OLD version of AIPS.  For NRAO  this  is  a  frozen  version
        which has been distributed worldwide.

new     Start  the  NEW  version  of AIPS.  For NRAO this is the most recently
        released version and is frozen right at the  time  of  initial  public
        release.

tst     Start  the  TST  version  of  AIPS.   For NRAO this is the unreleased,
        development version.  This is the default.

TV=[tvdisp][:][tvhost] or TV=local[:n]
        TV display server to use instead of the  default.   The  AIPS  startup
        script  tries  to  deduce  which  host the user is sitting in front of
        (this may not work; it is often difficult or impossible  to  determine
        this  information).   This may not be the same as the machine on which
        AIPS is to be run if, for example, the user has remotely logged in  to
        another machine within a terminal emulator window.

        The  "TV=local" option allows use of Unix based sockets for the TV and
        other servers.  If you choose this option, you MUST run the XAS server
        and  any  AIPS  sessions that will use it on the same host, though the
        DISPLAYs can be the same or different.  Also, no remote AIPS  sessions
        will be able to talk to this local TV.

        If  you  instead  use  "TV=local:0",  it  will  attempt to start a new
        instance of the TV and ancillary servers.  This can be  used  to  have
        multiple  TV's  on  the  same  host, and is useful in a compute server
        environment with X terminals.  If you have multiple Unix-socket  based
        TV's already started, you can choose which one a new AIPS session will
        use by, e.g. "TV=local:2" to choose the second one.

        NOTE: The default TV behavior is to use  INET or Internet based  sock-
        ets,  as  the  scripts  have  been doing since 1992.  The "local" Unix
        socket based functionality does not change this.

        For the default use of internet sockets, the full syntax  of  the  TV=
        option is TV=tvdisp:tvhost, where tvhost is the name of the machine on
        which the TV display server (usually XAS), Tektronix  graphics  server
        (TEKSRV),  message server (MSGSRV), and TV Lock server (TVSERV) are to
        run, and tvdisp indicates the machine to which the DISPLAY environment
        variable  should point for XAS.  Do NOT specify TV=hostname:0.0!  Both
        TVHOST and TVDISP can be different from the machine that  AIPS  itself
        is  running  on.  See the section on X Window System servers below for
        more information on how to control the servers.

        The default behavior of this option  if only one of tvdisp and  tvhost
        is specified is

        TV=tvhost   tvdisp defaults to tvhost.
```

```
         TV=tvdisp:  tvhost defaults to the host AIPS is running on.

         TV=:tvhost  tvdisp defaults to the host AIPS is running on.

         For  the remote TV options to work, you must be able to use the rsh or
         remsh command; see the notes on it under the tp= heading below.   Also
         see  the notes on environment variable AIPSREMOTE.  By default, if you
         do not specify any tv= option, you will only get a TV if your  current
         TERM  environment  variable matches sun*, *xterm*, *hpterm, dtterm, or
         iris*.  The DISPLAY environment variable is used if set, otherwise the
         who  am  i  (on  HP-UX, with the -R option) is used to make a guess at
         "where" you really are.

NOTV     Prevents automatic activation of the  TV  servers  if  no  display  is
         wanted.   This option also disables the Tektronix graphics server, the
         message server and the TV lock server.  See the section  on  X  Window
         System  servers  below for information on how to control the Tektronix
         and message servers.

TVOK     Assume that the TV display servers are already running;  the  particu-
         lars  (display, host) are still worked out -- from the TV=... argument
         (see above) if necessary -- but no servers will be started.

DA=host[,host,...] or DA=default or DA=all
         Select user data areas (directories, or "disks" in AIPSpeak) that  are
         local  to  the  (comma  separated)  list of machines.  Data areas from
         "required" hosts and those on the  local  machine  are  always  added,
         regardless of the list of hosts.

         All disks from each named host will be assigned.  Use the FREE command
         within AIPS to see the disk assignments you end  up  with.   They  are
         also shown on startup.

         AIPS  has  a  limit  of 71 disks in any one session.  The limit on the
         number of disks that can be defined for any given site is 512.  Disk 1
         is special in that it stores the AIPS message and save/get files.  The
         system is designed so that one particular required  disk  will  almost
         always  be  assigned  as disk 1.  For performance reasons, this may be
         undesirable if the filesystem in question is mounted via NFS.  See the
         description of personal .dadevs files below, as it can be used to cus-
         tomize the list of possible user data areas.

         Selecting DA=ALL will try to include every area defined in the startup
         file, up to the session limit.  Bear in mind that most AIPS tasks only
         have 10 slots for  "BADDISK".   Selecting  DA=DEFAULT  will completely
         bypass the configurable data areas  and  choose only  those data areas
         pre-configured by the AIPS manager; THIS IS NOT NORMALLY ENABLED, CHECK
         WITH YOUR AIPS MANAGER BEFORE USING DA=DEFAULT.

         There  is  a hierarchy of data area "lists" that AIPS will look for on
         startup.  These are:

         $HOME/.dadevs       This would be in your  private  login  area  (what
```

$HOME points to). It need not exist. If it doesn't, AIPS looks for the next file:

$DA00/DADEVS.LIST This is a host-specific file possibly set up by the AIPS manager. If it doesn't exist, AIPS finally looks for:

$NET0/DADEVS.LIST which is the site-wide data area configuration file.

The normal state of affairs is to have just one place for disks to be defined, namely $NET0/DADEVS.LIST. Your AIPS manager can choose to install host-specific list files, and you can choose (if you run AIPS from your own private account) to override both of these two with your own private version. This allows for considerable flexibility but moves the onus of maintenance of these files to the user. In other words, if you have your own .dadevs file, you have to keep track of your site's disk configuration!

If your AIPS installation supports multiple sites, e.g. to support both little-endian (Intel, Alpha) and big-endian (Sparc) systems, you can have any of these files refer to one or the othe site by appending the site name, e.g. $HOME/.dadevs.VCOARN for SITE=VCOARN.

The format for these files is all the same: a list of directory names preceded by a "+" for required or a "-" for optional. There should be two (2) spaces between the "+" or "-" (in the leftmost column) and the directory name.

In addition to all of the above, you may define a list of data areas in an optional $HOME/.dadevs.always file. This is used in addition to whichever of the DADEVS files have been selected by the rules above. The data areas that you will be assigned start with any required data areas in the $HOME/.dadevs.always file followed by any required data areas in the selected DADEVS file regardless of name. The 3rd group of data areas are those optional ones in the selected DADEVS file containing a string matching the user's host name. Then come the optional data areas in the $HOME/.dadevs.always file regardless of name. Finally, the optional data areas in the selected DADEVS file with strings matching the names of any hosts given in the DA=host[,host,...] command-line option. The order of data areas within each group, i.e. which is disk 1, etc., is determined by the order in the files.

There is also a $NET0/NETSP file that is maintained by the AIPS manager and controls aips user-number access to the disks. You will get error messages if your private .dadevs file includes AIPS data areas ("disks") that are not in the NETSP file. Regardless of the number of sites in your installation, there is only one NETSP file.

PR=# Select printer number (e.g., pr=2). If this option is not specified, the user will be presented with a menu of available printers and prompted to enter a choice. If there is only one printer configured, no menu will be presented. You may change the selected printer within

```
        AIPS via the PRINTER adverb.

REMOTE or REM or TEK
        Any  one  of  these indicates that the user is running from a terminal
        with Tektronix display  capability.  Graphics  output  will  be  sent
        directly  to  this  terminal.  NOTE: AIPS will not switch from text to
        graphics mode on terminals with a separate graphics "screen".

DEBUG[=prog][:aips]
        Start AIPS in debug  mode.  With  no  arguments,  the  user  will  be
        prompted  for  the name of the debugger (e.g. gdb, dbx, adb, csd, xde,
        dbxtool, debugger, xxgdb) and also whether to run  AIPS  itself  under
        the  debugger.  If you answer no, only AIPS tasks will be run in debug
        mode.  If =prog is specified, this suppresses the prompt for the  name
        of  the  debugger program.  If :aips is specified, this suppresses the
        prompt for whether to run AIPS itself in debug  mode  and  assumes  it
        will.   Use of both these options is useful in speeding up the startup
        of the system when debugging a program or AIPS itself.

LOCAL   Start a local copy of AIPS.EXE  residing  in  the  current  directory.
        Usually used by programmers for debugging purposes.

NORL    Disable  GNU  readline library and command-line editing.  This is pri-
        marily useful for running backgrounded  AIPS  sessions,  running  AIPS
        from "here-document" shell-scripts, and for debugging.

NOEX    This defers AIPS execution and is not normally used directly by users.


X WINDOW SYSTEM SERVERS
       If  you are running under the X Window System, AIPS will open up to three win-
       dows: a TV window (normally XAS), a message window  (MSGSRV)  and  a  graphics
       window  (TEKSRV).  If you specify the notv option on the command line, none of
       these will be started.

       MSGSRV and TEKSRV are actually simple programs running inside a terminal  emu-
       lator.   You  may use any terminal emulator that you would normally use on the
       machine on which you are running AIPS for the MSGSRV window.  Examples include
       xterm  (the  sample  vt100/Tektronix emulator that comes with the MIT X Window
       System code); cmdtool and shelltool (the standard terminal emulators for Open-
       Windows)  and  AIXterm  (the standard terminal emulator on RS/6000s).  You can
       choose which one to use by setting the environment variable  AIPS_MSG_EMULATOR
       to  the  name  of  the terminal emulator you wish to use.  For example, if you
       want to use cmdtool you would type

       setenv AIPS_MSG_EMULATOR cmdtool

       if you use the C or TC Shell, or

       AIPS_MSG_EMULATOR=cmdtool; export AIPS_MSG_EMULATOR

       if you use Korn, BASH, or Bourne shells before you start up AIPS.   You  could
       also   add   these   commands   to   your   .login  file  (C  Shell)  or  .profile
```

```
       (Korn/BASH/Bourne Shells) to make the assignment more permanent.  You can also
       give AIPS_MSG_EMULATOR the special value of "none" which will disable the mes-
       sage  window  without  affecting  the  Tektronix  window  or   the   TV.    If
       AIPS_MSG_EMULATOR is not set, the default is xterm.

       You  may  choose the terminal emulator used for the Tektronix window using the
       environment  variable  AIPS_TEK_EMULATOR  in  the  same  way  that   you   use
       AIPS_MSG_EMULATOR  to  choose  the terminal emulator, but it must support Tek-
       tronix graphics codes.  On most machines the only values of  AIPS_TEK_EMULATOR
       that  make any sense are xterm and none.  If AIPS_TEK_EMULATOR is not set AIPS
       will behave as if it were set to xterm.  (Note: dxterm, aixterm,  and  cmdtool
       are not "xterm"; they cannot display Tek graphics).

       You  can set preferences for positions and  colors for all three servers using
       the standard X Window System mechanisms.  Further  information  is  available
       through the AIPS HELP system (subjects MSGSRV, TEKSRV, XAS and XVSS).

       Note  that  AIPS  expects that a terminal emulator can start a program that is
       specified using a -e flag on the command line.  This is true  of  all  of  the
       terminal emulators we know about but if you find one that requires a different
       flag you can specify the flag as AIPS_TEK_EXE_FLAG or AIPS_MSG_EXE_FLAG.

ENVIRONMENT VARIABLES
       In addition to the Message and Tek server customizations, you  may  choose  to
       set a variable AIPSREMOTE to indicate your choice of remote shell command.  It
       is strongly recommended that the secure shell (ssh) be used in  place  of  the
       traditional Berkeley rsh or remsh command:
              setenv AIPSREMOTE "ssh -n"
       for csh or tcsh shells, or
              export AIPSREMOTE="ssh -n"
       for bash, korn, zsh and other bourne-like shells.
```

If you do not specify a printer (by number) on the command line when starting AIPS, you will get a menu showing you all the alternative printers available. You should omit the PR option until you are familiar with the choices. The OLD version of *AIPS* is likely to be relatively free of bugs (provided the *AIPS* version in NEW does not prescribe format changes which prevent OLD from working), but the NEW version will contain improvements and will be mostly debugged. The TST version is a debugging area recommended for NRAO staff and those few users who may require the most recent software. (Note that this choice affects only the version of the AIPS program itself. You may choose TST, NEW or OLD versions of the *AIPS* reduction programs at a later time — see § 3.5.)

2.2.4 Typing commands to the AIPS program

As of the 15JUL95 release, *AIPS* is available to users under a GNU-style license. This has numerous benefits, one of which is that it allows us to incorporate other GNU-style code within our system. The first of these is the GNU `readline` library which provides the user-input interface for AIPS under Unix beginning with the 15JAN96 release. The GNU readline library gives the user the ability to use the cursor-arrow keys, as well as various "control" and "escape" key sequences, to recall previously-entered commands, to edit the current command line (without having to back-space and re-type the entire line), to search the command history for previously-executed commands, to define customized key bindings for executing commands and macros, and much more. The full information may be obtained with the command `man readline` from the system command line (not inside AIPS). There is even "tab completion" based on the list of *AIPS* help files and on context. At any point, when typing a symbol, you may hit the TAB key. The symbol name will be completed if it is unique or the screen will flash (or the bell sound) is it is not. A second hit on the TAB key will produce a list of the possible completions. Since a task name cannot be the first symbol on a line, tasks are included in the possible completions only after some other symbol appears on the line.

The default key bindings should be very familiar to users of emacs and/or the bash shell; many of them should also be recognizable to users of the Korn and tcsh shells. Hard-core vi users can put AIPS into "vi-mode" and use vi-like key bindings instead. (The basic emacs-like key bindings will be outlined below; it will be assumed that those who are using the non-default vi-like key bindings already know what they are doing.)

Your command-line history is automatically saved between sessions, unique to both the user number and the "*AIPS* number" of the session, and then recovered at the next AIPS startup.

Use of the GNU readline library for input can be disabled on a per-session basis by starting AIPS with the "norl" option. This can prevent problems under some operating systems (most notably HP/UX) with putting AIPS into the background, when running with input "fed" from a script, or when debugging AIPS itself.

The key bindings are given below. Key sequences/bindings using the CONTROL key will be prefixed below with "C-." Those using the ESCAPE key (or "META" key — often available as the ALT key on PC keyboards and as the "diamond" key on Sun keyboards) will be prefixed with "M-." The basic cursor-movement key bindings are:

```
C-b     backward-character      [also: left-arrow]
C-f     forward-character       [also: right-arrow]
C-p     previous-command        [also: up-arrow]
C-n     next-command            [also: down-arrow]
M-b     backward-word
M-f     forward-word
C-a     beginning-of-line
C-e     end-of-line
C-r     incremental-search backward
C-p     previous-history (move backward in history list)
C-n     next-history (move forward in history list)
```

The basic editing key bindings are:

```
C-d     delete-character (under cursor)
M-d     delete-word (to right of cursor)
M-DEL   delete-word (to left of cursor)
C-t     transpose-characters (left with under cursor)
```

DELETE and BACKSPACE work as expected.

2.3 Managing windows

Unfortunately, the management of windows on a workstation screen depends heavily on the type of window manager and on the setup files defined for your login. At best, we can only be approximate here and try to describe general characteristics of normal setups.

2.3.1 General window management

Most window managers allow multiple windows to be created on the screen at the same time. These windows can either be closed in a small "iconified" form or opened in a larger and more usable form. Windows are normally opened by positioning the cursor on the icon with the mouse and clicking either once or twice with the left button. You can type only into open windows. An open window can be re-sized usually by "grabbing" (position the cursor with the mouse and then hold down the mouse button) one of its corners with the left mouse button. Windows under `twm` have a widget in the upper right corner which must be grabbed with any of the buttons. Positioning the cursor in the top bar of a window border and holding down a mouse button will do something. Usually, the left button moves the window, the middle button puts the window above or below other windows, and the right button gets you a pull-down menu of all the window manipulation options. Under `Motif` the middle and right buttons are switched. In the upper left corner of the top bar is a special button widget. Under `Openlook` and `twm`, clicking on this widget iconifies the window. Under `Motif` the iconify widget is shown as a dot and is usually in the upper right corner. The widget in the upper left corner under `Motif` offers a pull-down menu of window options, but is dangerous since a double click on that widget with the left button destroys the window (and any programs running in it).

Positioning the cursor in the root window (the background) and holding down a mouse button usually gets a pull-down menu with programs that can be run and various other options including exiting from the system. Well-configured systems offer a separate menu with each button. This is usually the way to get more windows if you need them.

When encountering a system for the first time, you should explore what the various controls have to offer. Position in the background, press each button in turn, and follow up what is offered by the pull-down menus which appear. Many menu items may themselves have a menu which you get by dragging the cursor to the right. Usually there is an arrow at the right of the menu item to indicate this. Then, try to open some icons with a single click or a quick double click. Then try the various mouse buttons in the top bar, the corners, and any special widgets visible in the window. Some windows also have scroll bars along the left or right side. Experiment with the various mouse buttons, clicking or dragging, in the scroll-bar region to see how to scroll back to previous text or forward to the last line. It pays to master all this slight of hand to allow you rapid access to multiple windows, previous text, and the like. It is very painful to click the wrong button and destroy a program that has been running for a few hours already!

2.3.2 Managing the *AIPS* TV window called XAS

On workstations, *AIPS* simulates a real TV display with a program called `XAS`. The program starts when you start `AIPS` and comes up in an open form. Its icon shows a cute drawing of an ape along with words like `AIPSTV UNIX` for local display and `AIPSTV INET` for Internet-connectivity. In many ways, this is a normal window which can be re-sized, moved, iconified, and destroyed like any other. However, when the window is open and the cursor positioned inside the window, `XAS` offers some additional features. The cursor changes shape and color in the window to indicate this fact. To get `XAS` to treat the cursor position as a "TV cursor" position, you must hold down the left mouse button. This allows the cursor to fill two rôles at nearly the same time, that of a workstation cursor and of a TV cursor. You do not have to hold the button down for long to register a TV position and, in fact, it is more efficient in interactive TV operations simply to click the left button at the desired locations. When you drag the cursor, numerous intermediate values are read with consequent extra computation. Note that the TV cursor position is read by `XAS` whenever the cursor is in the `XAS` window with the left button down. However, that position is only used when some verb or task reads it from `XAS` and uses it for some purpose, *e.g.*, to select image coordinates or to control image enhancement.

AIPS TV functions refer to "buttons" `A`, `B`, `C`, and `D` for the purpose of signaling conditions to the software. In the `XAS` simulation, these buttons are the keys a, A, or F3 for `button A`, keys b, B, or F4 for `button B`, keys c, C, or F5 for `button C`, and keys d, D, or F6 for `button D`. The F2 and F7 buttons toggle the size of the display from full screen to whatever size you set the window. `XAS` simulates a TV with a number, usually sixteen, of grey-scale memories and eight one-bit graphics overlays. The x and y dimensions of the memories adapt to the display area of your workstation less some room for window borders and, sometimes, for a few lines of a type-in window as well. `XAS` has the ability to display full-color (256 levels for each of red, green, and blue) on terminals capable of supporting full "TrueColor visuals." The internal dynamic range of images in `XAS` is actually 0:8191, but displays are limited to 0:255. You pay for this capability with a small reduction in speed for ordinary enhancements, blinks, and the like. You may select to limit your `XAS` to a "PseudoColor visual" which is all that is available on some older workstations. In that mode, the higher the number of grey levels the greater the dynamic range is available in the display of images. The maximum allowed maximum grey level is 235, but this will use all 256 levels of a "colormap" and therefore force `XAS` to use its own colormap. When the cursor enters the `XAS` window, the computer switches to that special colormap changing all of the other colors in the other windows (often in ways that are very undesirable). The default number of grey levels is 199 which may be small enough to avoid this effect or to manage to leave the colors of your most basic windows unaffected. Type HELP XAS C_R when in `AIPS` to see how to control the number of levels, the colors of the graphics overlay planes, and numerous other parameters.

2.4 Additional recipes

2.4.1 Banana daiquiri

1. Combine in an electric blender: 2 ounce **light rum**, 0.5 ounce **banana liqueur**, 0.5 ounce **lime juice**, 1/2 small **banana** peeled and coarsely chopped, and 1/2 cup crushed **ice**.
2. Blend at high speed until smooth.
3. Pour into large saucer champagne (or similar) glass. Serves one.

2.4.2 Banana pick-me-up

1. Slice ripe, peeled **bananas** into 3 cm chunks.
2. Wrap each chunk in strip blanched **bacon**.
3. Prepare mixture of **brown sugar** and **cinnamon** to taste.
4. Sprinkle mixture over banana chunks.
5. Bake at 350° F until the bacon is crisp and the sugar slightly caramelized.