Part II: Archive Retrieval Tool: arr Programmer's Guide

Carl Timmer (timmer@atdiv.lanl.gov)

EPICS R3.12

February 10, 1995

EPICS software was produced under U.S. Government contracts:
(W-7405-ENG-36) at the Los Alamos National Laboratory, and
(W-31-109-ENG-38) at Argonne National Laboratory.

Initially developed by:
The Controls and Automation Group (AOT-8)
Ground Test Accelerator
Accelerator Technology Division
Los Alamos National Laboratory

Co-developed with:
The Controls and Computing Group
Accelerator Systems Division
Advanced Photon Source
Argonne National Laboratory

CHAPTER 1 Tcl/Tk Installation

Disclaimer

This document is *NOT* intended to be a tutorial on tcl/tk. There are references given later on in this chapter for that purpose. In the later chapters, the author assumes the reader has some basic understanding of this software.

Before one can use **arr**, there are a few software packages that must be installed on the network somewhere. The main package is, of course, tcl/tk (pronounced tickle-tee-kay), but in addition to that the BLT extension package and the MListbox-patch are also required. An optional package is XF which is an interactive GUI-builder and allows modification of arr fairly painlessly. As far as versions are concerned, get tk3.6, tcl7.3, xf2.3, and blt-1.7. **Note, it is easiest to place all of these packages in the same directory** even though it is not necessary. This chapter shows how these packages can be obtained.

1.1 Acquisition of Tcl/Tk and Extensions

1.1.1 Acquisition of tcl/tk

The best way to obtain tcl/tk is by anonymous ftp. Previously (and perhaps again in the future), the best site to use was "harbor.ecn.purdue.edu". Currently (2/10/95) a good place to get tcl/tk is "ftp.cs.berkeley.edu" The transaction goes as follows:

```
% ftp ftp.cs.berkeley.edu
Name (ftp.cs.berkeley.edu:timmer): anonymous
Password: <user's email address>
ftp> cd ucb/tcl
ftp> bin
ftp> get tcl7.3.tar.Z
ftp> get tk3.6.tar.Z
```

Both files that are retrieved must be uncompressed and then untarred in the following manner:

```
% uncompress tcl7.3.tar.Z
% tar -xfB tcl7.3.tar
```

These actions result in the directories **tcl7.3** and **tk3.6** being formed. Both of these directories contain “README” files which the reader or the computer network administrator can digest and use to set up a tcl/tk installation directory.

1.1.2 Acquisition of the BLT Extension and Listbox Patch

The listbox widget of tk does not allow disjointed selections. However, a convenient patch to the listbox widget code will allow such selections.

The BLT extensions which **arr** uses are the graph widget, the ability to block input to windows while displaying a “timeout” cursor, and the ability to raise and lower windows.

Both of these items can be obtained by anonymous ftp from “ftp.aud.alcatel.com” as in:

```
% ftp ftp.aud.alcatel.com
Name (ftp.aud.alcatel.com:timmer): anonymous
Password: <user's email address>
ftp> cd tcl/extensions
ftp> bin
ftp> get MListbox.README
ftp> get MListbox.gZ
ftp> get BLT-1.7.README[1]
ftp> get BLT-1.7.tar.gZ[1]
```

As before, one must unpack the compressed files. In this case the commands are:

```
% gzip -d  MListbox.gZ
% gzip -d  BLT-1.7.tar.gZ
% tar -xfB BLT-1.7.tar
```

Heed the instructions of the README files. First apply the patch according to the instructions in the **MListbox.README** file. Unfortunately, the instructions are not very complete. To actually apply the patch, put **MListbox-patch** into the **tk3.6** directory and do the following:

```
% patch < MListbox-patch
.
.
.
File to patch:  doc/listbox.n          (type in prompted file name)
File to patch:  library/listbox.tcl    (type in prompted file name)
```

1. Note that newer versions of BLT may be available and hence the name of this file may change as well. To find the new name type `ls` after the `bin` command.

File to patch: `library/tk.tcl` (type in prompted file name)

When things are successfully completed and the final `make install` has been done, everything is ready to go.

1.1.3 BLT Bug Fix

At the time of this writing (version blt-1.7), there is a bug in the postscript output from the graph widget which can be traced to line 967 in **bltGrPS.c** in the **blt-1.7** directory:

```
"%d %d lineto\nDashesProc stroke\n\newpath %d %$d moveto\n"
```

but should be:

```
"%d %d lineto\nDashesProc stroke\nnewpath %d %$d moveto\n"
```

Another `make install` should apply the change permanently.

1.1.4 Acquisition of XF

This interactive GUI-builder can be anonymously ftp'd as well.

% `ftp ftp.aud.alcatel.com`
Name (ftp.aud.alcatel.com:timmer): `anonymous`
Password: `<user's email address>`
ftp> `cd tcl/code`
ftp> `bin`
ftp> `get xf-doc-us.ps.gz`
ftp> `get xf2.3-pl1.tar.gz`

These files must be unzipped and untarred:

```
% gzip -d  xf-doc-us.ps.gz
% gzip -d  xf2.3-pl1.tar.gZ
% tar -xfB xf2.3-pl1.tar
```

Follow the instructions of the **README** file that will appear upon following the above procedures. One may want to grab the **xf-doc.ps.gz** file instead of the **xf-doc-us.ps.gz** file if the reader would like the European format instead of the US format. File names are subject to change without notice.

1.2 World Wide Web

For those readers that have access to the internet through world wide web, the tcl/tk home page is "http://www.sco.com/IXI/of_interest/tcl/Tcl.html". This reference contains tcl/tk: FAQs, man pages, newsgroup information, applications, ftp sites, services, useful name & addresses, and the like. An alternative to using anonymous ftp to acquire tcl/tk and extensions is to use this web document to do all the work for you.

For "A brief introduction to TCL/TK" try "http://http2.brunel.ac.uk:8080/~csstddm/TCL2/TCL2.html". And for "Teacher Hypertools for TK/TCL" see "http://www.ece.cmu.edu/afs/ece/usr/svoboda/www/th/homepage.html". If you want the latest information on TCL 4.0 from John Ousterhout at Sun Microsystems, see "http://playground.sun.com/~ouster/#reporting". For additional, complex, Motif-like tk widgets implemented from basic tk widgets, check out **Tix** available on the web at "http://www.cis.upenn.edu/~ioi/tix/tix.html".

1.3 Newsgroup

There is a newsgroup called "comp.lang.tcl" which one may use to ask questions of tcl/tk experts around the world. One may also monitor the newsgroup's conversation if one does not have the means to use the newsgroup. Send the message "subscribe tcltk" to "listserv@cernvm.cern.ch". To stop receiving this enormous amount of mail, send "signoff tcltk", and for more list server commands send "info refcard".

For those with VMS operating systems, there's a tcl mailing list: vms-tcl@src.honeywell.com . An archive for the mailing list is available at: ftp://src.honeywell.com/pub/vms-tcl/mailinglist.archive .

1.4 Reference Literature

The main reference all ticklers use is Dr. John K. Ousterhout's book, Tcl and the Tk Toolkit published by Addison-Wesley (ISSBN 0-201-63337-X). If the outrageous price of this book is too much, the reader can acquire an earlier, free, postscript version by anonymous ftp from the same place tcl/tk is located in the files **book.p1.ps.Z** to **book.p4.ps.Z** .

Recently, a former student of Ousterhout's, Brent Welch, has posted early versions of a book he's writing on Tcl/Tk. He's gives lots of neat little tricks on programming. This draft can be ftp'd from "parcftp.xerox.com". Change directories to "pub/sprite/welch" and grab **tclbook.p**s or **tclbook.ps.Z** .

Man pages are the next most useful references. These, of course, come with the tcl/tk distribution and can be printed by the user. One can also access man pages through the tcl/tk www document mentioned previously.

There is a tk application called **wtour** which is a nice, beginning-level tour of all the widgets. It's available through anonymous ftp at harbor.ecn.purdue.ecn in the pub/tcl/code directory under the name of **wtour2.0.tar.gz** .

CHAPTER 2 arr Installation

Once tcl/tk is on the computer network, the installer can begin to worry about installing the archive retrieval tool. In addition to tcl/tk and related packages, there is a compressed tar file that must be acquired from LANL. The tar file, **arrtk.tar.Z**, when uncompressed and untarred, contains everything necessary for a working version of **arr**. Let the installer be aware that as of the present, **arr** has only been tested on a SUN IPC workstation running SunOs 4.1.3 . Please report any problems to Carl Timmer (ctimmer@lanl.gov).

2.1 Application Directory

The archive retrieval application is setup as essentially a custom version of tk. What makes it different from plain tk is that there are six commands that have been added for the reading of archive data files and handling of timestamps. The blt widgets have been added as well.

To uncompress and untar the application file, it is best to change directories to the one containing the tk, tcl, xf, and blt main directories. Place **arrtk.tar.Z** in that directory, and simply type the following:

```
% uncompress arrtk.tar.Z
% tar -xfB arrtk.tar
```

This action produces the directory **arr_tk** containing the following files:

1.	arr	(custom tk script)
2.	arr_tk.c	(custom C code)
3.	arrAppInit.c	(modified tkAppInit.c)
4.	configure	(GNU makefile configuration script)

5.	configure.info	(GNU information on configure)
6.	Makefile.in	(GNU makefile template)
7.	README	(instructions on application setup)
8.	arrUsersGuide.ps	(instructions on arr usege)
9.	arrProgrammersGuide.ps	(instructions on arr usege)

2.1.1 arr

This file is a tcl/tk script. Programmers are welcome to change any part of **arr** to suit their specific needs. The interactive GUI-builder by the name of **X**F may be used to do any modifications. See Chapter 1 about obtaining of copy of **XF**.

2.1.2 arr_tk.c

This file contains the modified versions of original archiver subroutines that have been converted into tk commands. It is written in traditional C and should require no modifications.

2.1.3 arrAppInit.c

This file is a modified version of tkAppInit.c which registers user C code as new tk commands. Read Tcl and the Tk Toolkit by Ousterhout for more information on it.

2.1.4 configure

This file is a GNU Makefile configuration script. It is used along with **Makefile.in** to produce an appropriate **Makefile**.

2.1.5 configure.info

This ascii file contains information on the **configure** and **config.status** files.

2.1.6 Makefile.in

This file is the template for **Makefile**.

2.1.7 README

This ascii file contains instructions on how to setup the **arr** application.

2.1.8 arrUsersGuide.ps

This postscript file contains general directions on the usage of **arr**.

2.1.9 arrProgrammersGuide.ps

This postscript file contains specific details concerning the usage of **arr**.

2.2 Application Compiling

Once the application directory is properly unbundled, there are three simple steps to follow to make things operational:

2.2.1 Configuring the Makefile

Type "./configure" in the **arrtk** directory to create the appropriate **Makefile**. This script searches (among other things) for tk, tcl, blt, and epics. If it cannot find what it needs, it will prompt the user for input as to the whereabouts of certain files and directories. If it is successful, it will create the files **Makefile**, **config.status** and **Fonts**.

2.2.2 Running the Makefile

If the first step completes successfully, type "make". This should create the files **arrwish**, **arrAppInit.o**, and **arr_tk.o** . The file **arrwish**, is similar to tk's **wish** with the added availability of the blt widgets and two commands for reading data from EPICS archive files.

Should the user change the makefile over time, the original can be recovered by running **config.status** . This script will change the makefile back to the state in which it first appeared upon running "./configure" .

2.2.3 Changing arr

Finally, one slight modification may be necessary to the first line of **arr**. This line reads**:**

```
#! arrwish -f
```

but should be changed to reflect the full pathname of the **arrwish** binary on the user's system as in

```
#!/users_actual_long_path_name/arrwish -f
```

If the length of this line is over 32 characters (full pathname over 27 characters), some systems will misbehave in confusing ways! To circumvent this limitation, change the first line to the form:

```
tail +2 $0 | /users_actual_long_path_name/arrwish $* ; exit
```

If this change is made, be aware that any further changes to **arr** using **xf** will cause **xf** to quit cooperating. Thus, for the sake of using **xf**, the first line must be temporarily changed back to the form "#!/path/arrwish -f".

2.3 Additional Changes

If any changes need to be made in **arr_tk.c**, the user only needs to type "make" to have them become permanent. Any changes to **arr** need no compilation as it is only a script.

CHAPTER 3 arr - epics interface

This chapter presents the reader with a look at the EPICS archiver subroutines and their implementation as new, specialized tk commands. The file **arr_tk.c** contains the code defining six new tk commands for use in a new version of the tk interpreter, **arrwish**. The first command, arr_fileList, returns a list all of the process variables in, the beginning and ending timestamps for, and the type of the selected archiver data file. The second command, arr_getData, collects and returns all relevant data (given a list of process variables and retrieval conditions). The other commands, arr_tsAddSec, arr_tsCmpStamps, arr_diffAsSec, and arr_diffAsStamp, are used to manipulate timestamps.

3.1 Archiver Subroutine Location

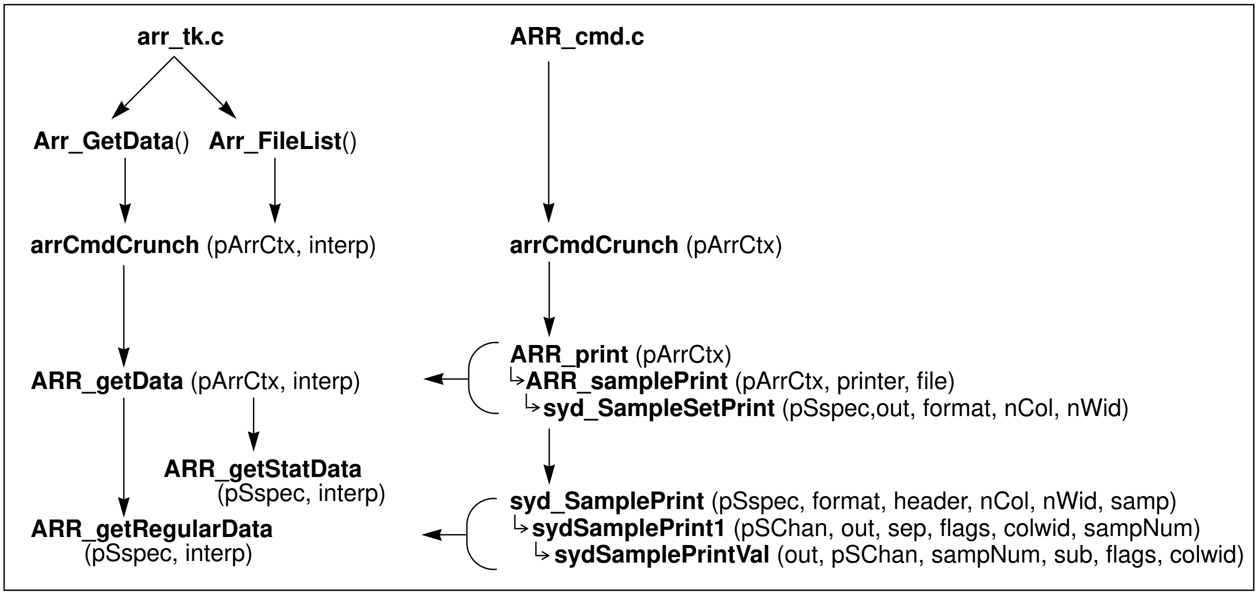
The archive retrieval tool's EPICS functionality is provided by the use of Roger Cole's archiver routines. The source code for these routines (pre R3.12) can be found in directories and files as in the following:

1. ~epics/r3.11/share/src/ar . All files are relevant. See especially ARR_cmd.c .
2. ~epics/r3.11/share/src/libCom . Look at ar*.c, bfSub.c, sydSubr*.c, and tsSubr.c .
3. ~epics/r3.11/share/epicsH . Look at ar*.h, bfDefs.h, syd*.h, and tsDefs.h .

3.2 arr_tk.c's Place of Origin

Subroutines in **arr_tk.c** were taken from **ARR_cmd.c** for the reason that they contained the basic functionality needed. They were non-graphical routines and thus were small and much easier to comprehend than other code. The routines from **ARR_cmd.c**, used to print out data, were modified to send this data to a tk interpreter. Most archiver errors were also turned into tcl/tk errors. In addition to those changes, the architecture was flattened in the following way:

FIGURE 1 calling sequences old (right) and new (left):



For those programmers interested in archiving software, routines beginning with the prefix "syd_" allow access to both binary and ascii archive data files and some channel access capabilities as well.

3.3 arr_fileList

NEW TK COMMAND

This new tk command, found in **arrwish**, is used as follows:

arr_fileList *pathname filename*

This function returns a tcl list containing (in this order) a tcl list of all process variables, the oldest timestamp, the newest timestamp, and the type of file which are found in the file given by *filename* located in the directory *pathname*. The timestamps are of the form, "03/24/93 09:03:09.000", and the file type is either "byChannel" or "sampleSet".

The following tcl code gives an example of how to use this command:

```
if {[catch "arr_fileList $path $file" returnMsg] == 1} {
  AlertBox "IGNORING REQUEST:\n$returnMsg" "" "200x100" "WARNING"
  return
} else {
  set channelNames [lindex $returnMsg 0]
  set oldTimestamp [lindex $returnMsg 1]
  set newTimestamp [lindex $returnMsg 2]
  set fileType     [lindex $returnMsg 3]
}
```

3.4 arr_getData

This new tk command, found in **arrwish**, is used as follows:

NEW TK COMMAND

arr_getData *filename filetype pvlist retrieveCode retrieveList [partialSamples] [mean] [roundStamps] [condition]*

The arguments are defined as follows:

filename	- the full name of the EPICS archive file
filetype	- either "sampleSet" or "byChannel"
*pvlis*t	- a tcl list of all channels for which data is desired
retrieveCode	- NUMBER+ if using starting timestamp & # of samples forward in time; NUMBER- if using starting timestamp & # of samples backward in time; TIMESTAMP if using beginning and ending timestamps to define retrieval
retrieveList	- a tcl list containing the earliest timestamp, the starting timestamp, and either the ending timestamp if the retrieveCode is TIMESTAMP or the # of samples if the retrieveCode is NUMBER+/-
partialSamples	- 1 if partial samples are to be discarded, else 0
mean	- 1 if using means and standard deviations, else 0
roundStamps	- round retrieved timestamps to this value (millisec). If 0, no rounding occurs.
condition	- logical retrieval condition. The form of *condition* must follow the format, `pvname operator value`, where `pvname` is the name of a process variable in the file of interest; `operator` is a relational operator such as >, >=, <, <=, ==, !=; and `value` is the value of the process variable.

This command returns a single, complex tcl list of all data for all process variables found in the file given by *filename*. That list, in turn, is comprised of the following elements in the order given:

(1) a tcl list of process variable (channel) names for which data has been retrieved;
(2) a tcl list of all the timestamps for which data has been retrieved in oldest to newest order;
(3) a tcl list of times in seconds, referenced from the beginning timestamp, corresponding to the timestamps in the previous list, (2);
(4) since each channel may or may not have data at a timestamp of the list given in (2), this element is a tcl list of tcl sublists where each sublist contains the timestamps for which there is data for each channel in list (1);
(5) a list of sublists where each sublist contains the times in seconds corresponding to the sublist of timestamps in list (4);
(6) a list of sublists where each sublist contains the values of the channels in list (1);
(7) a list of units for the channels in list (1);
(8) a list of minimum values for the channels in list (1);
(9) a list of maximum values for the channels in list (1);

(10) if statistics are being used, this is a list of sublists where each sublist contains the standard deviations corresponding to the values in the sublists of (6). If statistics are NOT being used, this is a list of C data types of the values of the channels in list (1). The types are string, double, float, long, or short.;

(11) if statistics are being used, this is a list of the number of samples used in the calculation of means and standard deviations at each timestamp given in list (2). If statistics are NOT being used, this is a list of sublists where each sublist contains the first status code of the values in the sublists of (6).;

(12) if statistics are NOT being used, this is a list of sublists where each sublist contains the second status code of the values in the sublists of (6). WHEW!!

The first 5 arguments are required, but the last 4 in brackets are optional. If one of the optional arguments are specified, all previous arguments must also be specified. Note that it is improper to pass a blank string for *condition*, and it will cause an error. Be careful to put quotes around any lists in the arguments or around any timestamps as they may contain white space.

The following tcl code gives an example of how to use this command:

```
if {"$condition" == ""} {
  if {[catch "arr_getData $fileName \
        $fileType \
        \"$pvNameList\" \
        $dataRetrieveCode \
        \"[list $earliestTimestamp $dataLoadStart $dataLoadStop]\" \
        $partialSample_flag \
        $meanStDev_flag \
        $roundTimestampsTo" returnList] == 1} {

      AlertBox "FAILURE TO GET DATA:\n$returnList"  ""  210x120 "WARNING"
      return
  }
} else {
  if {[catch "arr_getData $fileName \
        $fileType \
        \"$pvNameList\" \
        $dataRetrieveCode \
        \"[list $earliestTimestamp $dataLoadStart $dataLoadStop]\" \
        $partialSample_flag \
        $meanStDev_flag \
        $roundTimestampsTo \
        $condition" returnList] == 1} {

      AlertBox "FAILURE TO GET DATA:\n$returnList"  ""  210x120 "WARNING"
      return
  }
}
####################################################
# Decode the returned "returnList" into components #
####################################################

# simple lists
set pvNameList           [lindex $returnList 0]
set masterTimestampList  [lindex $returnList 1]
set masterTimeList       [lindex $returnList 2]

# lists of lists
```

```
set all_timestampLists  [lindex $returnList 3]
set all_timeLists       [lindex $returnList 4]
set all_valueLists      [lindex $returnList 5]

# arrays
set unitsList           [lindex $returnList 6]
set minimumList         [lindex $returnList 7]
set maximumList         [lindex $returnList 8]
set typeList            [lindex $returnList 9]

# for each channel we have one element in various arrays with its data
set index 0
foreach i $pvNameList {
  set units($i)         [lindex $unitsList   $index]
  set minimum($i)       [lindex $minimumList $index]
  set maximum($i)       [lindex $maximumList $index]
  set type($i)          [lindex $typeList    $index]
  set timestamps($i)    [lindex $all_timestampLists $index]
  set times($i)         [lindex $all_timeLists      $index]
  set values($i)        [lindex $all_valueLists     $index]
  incr index
}

# if using statistics, else ...
if {$usingStats_flag == 1} {
  set statCounts [lindex $returnList 10]
  set all_stdevs [lindex $returnList  9]
  set index 0
  foreach i $pvNameList {
      set stdevs($i) [lindex $all_stdevs $index]
      incr index
  }
} else {
  set all_dataStatus1 [lindex $returnList 10]
  set all_dataStatus2 [lindex $returnList 11]
  set index 0
  foreach i $pvNameList {
      set dataStatus1($i) [lindex $all_dataStatus1 $index]
      set dataStatus2($i) [lindex $all_dataStatus2 $index]
      incr index
  }
}
```

3.5 arr_diffAsSec

NEW TK COMMAND

This new tk command, found in **arrwish**, is used as follows:

arr_diffAsSec   *timestamp1 timestamp2*

This function returns a time in seconds equal to *(timestamp1 - timestamp2*). Timestamps are of the form, “03/24/93 09:03:09.000”.

3.6 arr_diffAsStamp

NEW TK COMMAND

This new tk command, found in **arrwish**, is used as follows:

arr_diffAsStamp   *timestamp1 timestamp2*

This function returns a timestamp equal to *(timestamp1 - timestamp2*). Sign is taken care of. Note that all time differences are referenced from a "zero" timestamp of 1/1/90 00:00:00.000 . So any time differences greater than 24 hours will be reflected in the date part of the timestamp relative to this reference date. Any time differences less than 24 hours will omit the date part of the returned timestamp. Timestamps are of the form, "03/24/93 09:03:09.000".

3.7 arr_tsCmpStamps

NEW TK COMMAND

This new tk command, found in **arrwish**, is used as follows:

arr_tsCmpStamps *timestamp1 timestamp2*

This function compares timestamps and returns:
-1 if *timestamp1* is earlier than *timestamp2*,
0 if *timestamp1* is the same as *timestamp2*,
+1 if *timestamp1* is later than *timestamp2*.
Timestamps are of the form, "03/24/93 09:03:09.000".

3.8 arr_tsAddSec

NEW TK COMMAND

This new tk command, found in **arrwish**, is used as follows:

arr_tsAddSec *timestamp timesec*

This function adds *timesec* number of seconds to *timestamp* and returns the resulting timestamp. Timestamps are of the form, "03/24/93 09:03:09.000".

CHAPTER 4 arr - tcl/tk interface

This chapter describes part of the script file **arr** which is written in tcl/tk.

4.1 User Modifications

There are a number of features that users may want to change to suit their own file structure or artistic taste. The text below provides a brief guide on how to make some of these changes, but in order to get a more complete picture the user should read Tcl and the Tk Toolkit by John Ousterhout.

4.1.1 arr Colors

There are two ways in which to modify the color scheme of **arr**. The first method requires the editing of **arr**. Close to the end of the file, there is a procedure called **StartupSrc**. In this procedure, color options are set in lines beginning with "option add". These lines can be edited (or new ones can be added) to provide the users with their own selection of colors.

The second method that can be used is to comment out the "option add" lines mentioned above and add lines to the users' **.Xdefaults** file. The syntax of lines such as:

option add Scrollbar.foreground LightBlue2

in **arr** become

arr*Scrollbar.foreground: LightBlue2

when added to the **.Xdefaults** file.

4.1.2 graph Colors

The selection of colors for graph elements is set in the list “colorList” near the end of the **arr** file under the heading “# initialize global variables #”. Likewise, the selection for the suggested graph background and border colors is set in “backgroundColorList”. Both can be easily changed. Make sure that any color names are also in the **rgb.txt** file.

4.1.3 Fonts

As in modifying **arr**’s colors, find the procedure called **StartupSrc**. Changing the “option add” font will change the main font. Change the font set by “labelFont” in order to change the font used in some of the labels. Use * and ? wildcards in font names to avoid having to specify every field. For example, the font name

    -helvetica-bold-r-normal---120-*

specifies a 12 point Helvetica font in normal width and height. Foundry, pixel size, and other parameters are ignored. If more than one font meet this condition, the X server will pick the first to match.

4.1.4 Directories

When selecting a data file, it is convenient to have the file selection window come up with a useful default data file directory. This can be set by modifying the line near the end of **arr** which starts with

    set {fsBox(path)}

to the directory of interest.