ADAPTOR
Installation Guide
Version 1.0 (June 1993)

T. Brandes

Internal Report No. Adaptor 1

June 30, 1993

High Performance Computing Center
German National Research Institute for Computer Science
P. O. Box 1316
D-5205 Sankt Augustin 1
Federal Republic of Germany
Tel.: +49 (0)2241 / 14-2492
E-mail: brandes@gmdzi.gmd.de

ADAPTOR
Installation Guide
Version 1.0 (June 1993)

T. Brandes

German National Research Center for Computer Science,
P.O. Box 1316, D-5205 Sankt Augustin 1, FRG

Abstract

ADAPTOR (Automatic DAta Parallelism TranslatOR) is a tool that transforms data parallel programs written in Fortran with array extensions, parallel loops, and layout directives to parallel programs with explicit message passing. The input language is very similar to CM Fortran and High Performance Fortran though not all features of these language are supported.

The generated message passing programs will run on different multiprocessor systems with distributed memory, but also on shared or virtual-shared memory architectures.

This paper describes how this tool will be installed for different parallel architectures and how the installation is tested.

1 Overview

Adaptor is a package that consists of:

- the interactive source to source transformation tool fadapt
- the distributed array library DALIB with communication routines and operations for distributed and local arrays
- documentation files in PostScript format,
- and many example programs.

Figure 1 shows that the tool `fadapt` generates message passing programs that will be linked together with the DALIB.

1.1 Installation Directory

Before downloading Adaptor you should create a directory for the installation. This installation directory will be referred to `PHOME`.

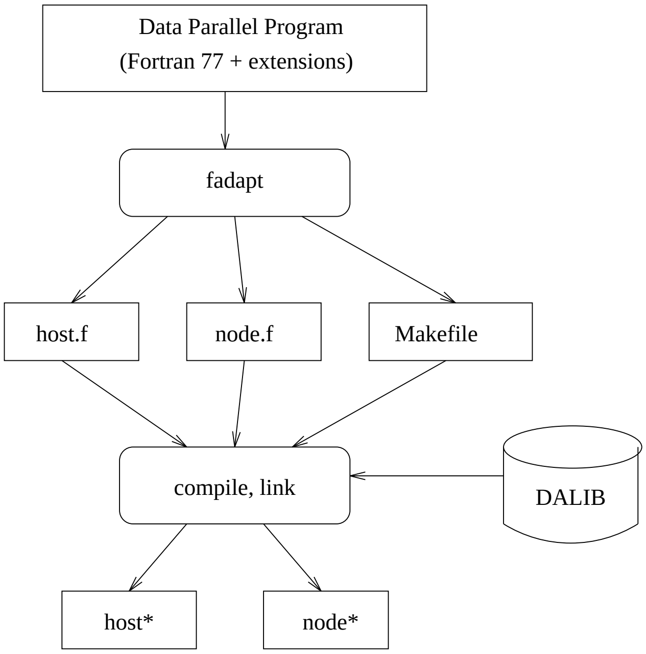

Figure 1: Overview of Adaptor

```
...
mkdir adaptor
cd adaptor
pwd         < this is PHOME >
$PHOME=`pwd`
```

When using Adaptor, every user should set a link to the Adaptor-directory in his home directory.

```
cd                        ! change to home directory
ln -s $PHOME adaptor      ! set a link to the installation directory
```

1.2 Downloading of Adaptor

Adaptor is available via anonymous ftp and should be downloaded directly in the installation directory.

```
cd $PHOME
ftp ftp.gmd.de   (129.26.8.90)
 user: anonymous
 password: <your e-mail address>
cd gmd/adaptor
binary
```

```
get adp_1.0.tar.Z
quit
```

At first the tar file has to be uncompressed before all the files of Adaptor can be extracted.

```
uncompress adp_1.0.tar.Z
tar xvf adp_1.0.tar
rm adp_1.0.tar
```

If no errors have occurred during extraction, the following files and subdirectories should be present in the Adaptor-directory:

```
help        (help file for interactive use)
welcome     (information file for interactive use)
bin         (directory for binaries)
src         (source files for fadapt)
examples    (directory with example programs)
docs        (documentation files in PostScript)
dalib       (libraries used for Adaptor)
man         (manual pages for Adaptor)
```

The installation of Adaptor should be proceeded in the following intended way:

1. Compile and link the transformation tool fadapt.
2. Test the transformation for some example programs.
3. Compile and install the DALIB for your parallel machine.
4. Compile and link some example programs and run them on your parallel machine.
5. Make some necessary updates for the transformation tool so that it fits into your installation.

2 Installation of fadapt

The most interesting part of Adaptor is the interactive source to source transformation. From a given data parallel source program (a description of this language can be found in the *Adaptor Language Reference Manual* [Bra93a]) an equivalent message passing program is generated.

The sources for the transformation tool are stored in the subdirectory **src**. The sources are all written in C and should run on all UNIX machines. The interactive version needs UNIX machines supporting X-Windows and the Athena widgets toolkit [NO90, O'R90].

The executable **fadapt** for the tool fadapt will be generated by simply typing make. Another executable **fstrip** is also generated that is needed to strip the new generated source files to a certain length.

```
cd $PHOME/src
make [batch]
mv fadapt ../bin
mv fstrip ../bin
make clean
```

Depending on the machine where the transformation tool is installed it might be necessary to make some changes in the Makefile (see section 2.2).

2.1 Information about the Source Files

Though most files of the source to source transformation have been generated with compiler tools, the tar file provides only the generated sources. Otherwise, it would be necessary to install the compiler tools in the correct version (the GMD compiler tools are also available in the same anonymous ftp area in the directory **gmd/cocktail**).

Therefore it is not recommended to make more complex changes of your own for the translation tool. Interested users who want to implement own transformations or to modify existing ones should install the compiler tools (GMD Karlsruhe Compiler Toolbox [GE90]) and ask the author for the corresponding compiler specifications.

2.2 Changes in the Makefile

The `Makefile` delivered for the installation of `fadapt` should work directly on SUN workstations. On other machines it might be necessary to make some changes for the variables. Especially the directory of X-Windows has to be known for the compilation.

Check the following variables within the `Makefile`:

- `CC` for your C-Compiler
- `CFLAGS` for compiler options
- `LFLAGS` for the link options

- `XDIR` for the directory where X-Windows is installed

These variables have to be set correctly. On some machines it might be necessary to increase the sizes of internal compiler tables (e.g. `-Wf,-XNg1500` must be included in `CFLAGS` for SGI machines).

2.3 Known Problems

The following problems may occur depending on the machine where the tool is installed:

1. When linking there are unresolved externals

   ```
   _get_wmShellWidgetClass
   _get_applicationShellWidgetClass
   ```

   These warnings can be ignored, but the execution bit has to be set explicitly.

   ```
   chmod +x fadapt
   ```

2. `R_OK` is not known in the files `fadapt.c` or `Scanner.c` or `makefiles.c`.

   ```
   fxc - error  : (1) R_OK is unknown in the current scope
   ```

   On the Alliant there has to be used the include file `fcntl.h` instead of `unistd.h`. Use the compiler switch `-Dalliant` in `CFLAGS`.

3. For the source files `Scanner.c` the redeclaration of `malloc` in yyInitialize has to be removed.

4. It could be necessary to undefine an environment variable for the compilation. Therefore insert in the file `Tree.h` in the subdirectory `include` the following line:

   ```
   #undef __STDC__
   ```

5. If Unix System V (e.g. on KSR 1, SGI) has problems with `xfiles.c`, the following change is needed:

   ```
   old: struct direct* direntry;
   new: struct dirent* direntry;
   ```

   This is done automatically by using the compiler switch `'-DSYS_V'`.

6. If the Athena widgets or X-Windows are not available, the batch version must be installed. This batch version is created in the following way:

   ```
   make batch
   ```

2.4 Predefined Options

Though all options for the translation can be changed via flags (see the em Users Guide of Adaptor [Bra93b]), some changes in the include file **global.h** in the directory **$PHOME/src/include** are useful to have the correct predefined options for your parallel machine.

```
#define predefined_target_machine  SUN4_PVM
#define predefined_target_language FORTRAN_77
#define predefined_target_model    HOST_NODE
#define predefined_array_kind      STATIC_ARRAYS
#define predefined_ddefault_kind   DDEFAULT_DISTRIBUTED

#define predefined_MinProc         1
#define predefined_StaticArraySize 25000
#define predefined_split_flag      0

#define predefined_PHOME           "/home/zdvex/zdv636/adaptor"
```

The value of `predefined_PHOME` must be the name of the installation directory. This value is very important as some files of this directory are used when calling **fadapt**.

Possible values for the predefined options can be found in the same file. The options above e.g. should be used to make Adaptor suitable for running parallel programs with PVM on a SUN4 workstation net.

Some further changes might be necessary for the file **makefiles.c** to guarantee that fadapt generates a correct Makefile that can be used directly on your parallel machine (see section 5).

2.5 Call of fadapt

After the successful compilation two executables named **fadapt** and **fstrip** have been generated. Move these files to the **bin** subdirectory of Adaptor.

```
mv fadapt ../bin
mv fstrip ../bin
```

You should include the **bin** subdirectory of Adaptor in your environment variable PATH.

```
setenv PATH $HOME/adaptor/bin:$PATH
setenv MANPATH $HOME/adaptor/bin:$MANPATH
```

2.6 Removing Object Files

After having successfully compiled type `make clean` in the directory where all the sources of `fadapt` reside. This will automatically delete all generated object files.

```
cd $(PHOME)/src
make clean
```

3 Testing of fadapt

For users with only small experience in running X-Windows applications should test the batch version of Adaptor at first.

3.1 Testing the Batch Use

```
cd $PHOME/examples/simple
```

The file `simple.f` can now be translated with the following command (this is possible with the interactive and the batch version, respectively):

```
fadapt -H simple.f
```

After the translation the following files should have been generated:

- `host.f` (host program),
- `node.f` (program running on all nodes),
- `Makefile`,
- the protocol files of the translation `adaptor.def`, `adaptor.sem`, `adaptor.cf`, `adaptor.anal`, `adaptor.dist`, `adaptor.temps`, `adaptor.init`, `adaptor.seq` and `adaptor.trans`

After the generation of the new source files it might be necessary for some machines to insert continuation lines in these programs as the source lines can be longer than 72 or 132 characters. Usually the following compiler error occurs:

```
Error on line 107 of cube.f: unbalanced parentheses, statement skipped
Error on line 107 of cube.f: Execution error unclassifiable statement
```

The striping can be done automatically with the command fstrip.

```
fstrip host.f 72
fstrip node.f [132]    /* 132 is default */
```

3.2 Testing the Interactive Version

The interactive version can only be tested if no problems with the X-Windows installation have arisen. Otherwise only the batch version can be used.

```
cd $PHOME/examples/simple
fadapt
```

After calling $\boxed{\text{fadapt}}$ a window should be displayed on your X-Server. If any problems occur, check whether

- the X-Server is running,
- the environment variable DISPLAY is set to the address of the machine running the X-Server,
- the client (in this case the machine running fadapt) must be authorized to write on the screen of the X-Server (use command xhost)

If your installation directory has not been set correctly in the file `global.h`, the following error message might appear:

```
No access on ADAPTOR Home-Directory ....
```

In this case you should make the necessary change and recompile the tool `fadapt`.

After the appearing of the window execute the following steps:

- Click with the mouse button on the $\boxed{\text{File}}$ command and select the file simple.f.
- Select the $\boxed{\text{Show}}$ command. The file is displayed in the editor area.
- Select the commands $\boxed{\text{Parse}}$ and then $\boxed{\text{Semantic}}$ for the transformation of the source file into an intermediate representation and for many checks.
- Select the command $\boxed{\text{Adapt}}$. After the translation a new window will be generated where the new sources are displayed.
- Select $\boxed{\text{Quit}}$ in the new generated window.
- Select $\boxed{\text{Quit}}$ in the main window, Adaptor should terminate.

These commands have the same results as the batch call described in section 3.1.

4 Installation of the DALIB

When the translation tool of Adaptor works correctly and first experiences have been made with the implemented transformations, the generated message passing programs should run on your parallel machine.

This is possible if the distributed array library DALIB is installed on your machine. This library is absolutely necessary for linking the parallel programs.

The DALIB has been realized in C. Most source files are machine-independent but every machine has some special machine-dependent realizations of communication functions and of the main programs that are responsible for the initialization.

4.1 Machine Directories

The dalib directory has the following subdirectories:

```
GENERAL        ! machine-independent files
PVM2.4         ! special files for PVM version 2.4
PVM3.1         ! special files for PVM version 3.1
ALLIANT        ! Alliant shared memory version
GC             ! Parsytec GCel, based on PARIX
IPSC           ! iPSC/860 version
MEIKO-CS1      ! version for Meiko CS1
MEIKO-CS2      ! version for Meiko CS2
CM5            ! CM-5 version using the CMMD message passing library
CM5a           ! same as CM-5 but using the CMF compiler for one node
X              ! display of pixel arrays using X-Windows
```

The subdirectory `GENERAL` contains the machine-independent source files. The subdirectories for the machines contain only the machine specific communication functions, the main programs for host and node processes and a makefile.

The main programs for the different processes are listed below:

```
mhost.c          ! main program of host process
mnode.c          ! main program of node process
mcube.c          ! main program of node process (without host)
mnode1.c         ! main program of single node process
```

For the shared memory architectures the file `mnode.c` does not exist as the host process forks itself.

After compiling the DALIB for one machine there should be the following files:

- `mhost.o` is the object file with main program that calls the host program and starts the node processes automatically

- `mnode.o` is the object file with main program that calls the node program
- `mcube.o` is the object file with main program that calls the node program when no host program is running
- `mnode1.o` is the object file with main program that calls the single node program
- `unilib.a` is the library file with some subroutines that are used for single node programs (timing, random numbers, array operations)
- `dalib.a` is the library file with subroutines for message passing, reductions, etc.

4.2 Modules of the UNILIB

These are the modules of unilib.a that are used for the single node version of the generated program:

```
timing0.c           ! timing for a specific machine
random.c            ! random numbers
memcopy.c           ! efficient memory copy
combiners.c         ! associative operations used for reductions/scatter

overlap0.c          ! overlapping of sequential arrays
cshift0.c           ! circular shifting of sequential arrays
schedule0.c         ! indirect addressing of sequential arrays
transpose0.c        ! transpose of sequential arrays
```

4.3 Modules of the DALIB

The modules of the unilib and the following ones are used together for the complete DALIB:

```
system.h           ! general include file
#
arguments.c         ! getting the number of node processes, trace flag
timing.c            ! timing for a specific machine
random.c            ! random numbers
mailbox.c           ! primitive send and recv with mailboxes
barrier.c           ! barrier synchronization
broadcast.c         ! broadcast from one node to all other nodes
reduction.c         ! reduction functions (sum, min, max, ...)
buffer.c            ! filling and sending communication buffers
manager1.c          ! management of array distributions
```

```
section1.c          ! help functions for send/recv of sections
hostnode1.c         ! transfer of arrays between host and node
replicate1.c        ! replication of distributed data
cshift1.c           ! circular shifting of distributed arrays
movement1.c         ! sending and receiving of distributed array sections
transpose1.c        ! transpose of distributed arrays
schedule1.c         ! indirect addressing of distributed arrays
trace.c             ! trace utilities
```

4.4 Installation of the Distributed Array Library

For the installation of the DALIB go into the subdirectory belonging to your machine and just type $\boxed{\text{make}}$.

```
cd $PHOME/dalib/<ARCH>
make
```

In the following some machine specific features are described.

4.4.1 PVM

PVM is a public domain software that can be installed on many different machines [Sun90]. At the moment DALIB has been realized upon this package for SUN4 and IBM RISC workstations. Also for the Alliant FX/2800 there is a version available.

When running the parallel program it is necessary that the PVM software is also correctly installed on your workstation net. This includes the library **libpvm.a** and the executable **pvmd** for the pvm daemon.

At the moment PVM version 2.4 and version 3.1 are supported.

PVM is also available on the ftp server at the GMD. The following commands will install you PVM 3.1 on your local workstation.

```
cd                                ! change to home directory
ftp ftp.gmd.de   (129.26.8.90)
 user: anonymous
 password: <your e-mail address>
 cd gmd/adaptor
 binary
 get pvm3.1.tar.Z                  ! download pvm
 quit
uncompress pvm3.1.tar.Z            ! uncompress
tar xvf pvm3.1.tar                 ! untar
rm pvm3.1.tar                      ! delete tar file
```

```
cd pvm3
make                                  ! hopefully it works
mkdir bin                             ! make some additional directories
mkdir bin/SUN4
setenv PATH $HOME/pvm3/lib/SUN4:$PATH
```

PVM 2.4 can be installed in the same way. Please note that there have been some essential changes from version 2.x to version 3.x.

4.4.2 Intel iPSC/860

The iPSC/860 needs two libraries as the front end system (SRM) running the host program has not the same processor as the nodes of the parallel machine.

```
cd $PHOME/dalib/iPSC/host
make
cd ../node
make
```

If a host program is used this will have to run on the SRM. Therefore the host library has to be compiled on the SRM in any case.

4.4.3 Meiko CS

The installation of the DALIB for the Meiko CS machine is similar to the installation on the iPSC/860. There are only some slight changes between the two versions. The realization on the Meiko machine uses a communication library that has the same functionality as the iPSC communication routines.

4.4.4 Parsytec GCel

The realization of the DALIB on the Parsytec GC series is based upon PARIX (version 1.0 or newer versions).

For broadcasts and reductions special functions have been implemented using a virtual tree topology.

Warnings during the compilation of the DALIB can be ignored.

Attention: with the current version of the DALIB for the Parsytec GC system problems with messages longer than 1 kByte might arise. A more sophisticated version is still in work.

4.4.5 Connection Machine CM-5

For the CM5 there are two versions available. Both versions are based on the CMMD 3.0. One version must be used with the SUN Fortran 77 compiler (subdirectory CM5), the other one with the CM Fortran compiler (subdirectory CM5a). There are some slight differences between the two versions for the initialization within the main programs.

4.4.6 KSR1

Though the KSR1 is a virtual shared memory architecture, the parallel processes will run as independent Unix processes that communicate with each other.

For the interprocessor communication the System V features **Shared Memory** and **Semaphores** are incorporated.

For an abnormal termination of a parallel program it is necessary to remove the created semaphore and the created shared region (using the commands `ipcs`, `ipcrm`).

4.4.7 SGI

The DALIB for the SGI multiprocessor systems is realized in the same fashion as for the KSR 1. The same Unix System V features are used.

4.4.8 Alliant FX/2800

On the Alliant (Alliant Concentrix 2800 2.2.00) the message passing of the DALIB has also been realized via a shared memory region. The used features are similar to the Unix System V features, but they have a slightly different syntax and semantic.

On the Alliant it is not necessary to remove created shared regions explicitly as it is required on the KSR 1 and on the SGI machines.

4.4.9 Other Machines

If you have a parallel machine that is not supported until now, please read the documentation of DALIB carefully in order to port one of the existent codes onto your machine.

Experiences have shown that only some files have to be changed, e.g. for basic communication routines (`mailbox.c`), broadcasting (`broadcast.c`), timing (`timing.c`) and for random numbers (`random.c`).

The example directory provides many programs that can be used for testing the implementation of the DALIB on a new machine.

4.5 Installation of the X-Windows library

In the subdirectory X some functions for X-Windows are implemented. This library is not really necessary hence it is required for some example programs. It realizes the display of two-dimensional arrays with pixels on a screen of X-Windows.

5 Testing the Compilation of Parallel Programs

After the distributed array library has been installed it should then be possible to compile and link the parallel program generated with the transformation tool.

Therefore switch to a directory where the generated files `host.f`, `node.f` and `Makefile` of a translation with Adaptor reside. Type $\boxed{\text{make}}$ and see what happens.

Hopefully, you have to make no or only small changes to the generated `Makefile`. As it depends on the target machine, at first check if the corresponding value is set correctly in the file `global.h` in the subdirectory `$PHOME/src/include`.

A typical makefile is the following one:

```
#
# BEWARE: this Makefile was automatically generated by Adaptor
#         Adaptor will overwrite any changes you make if you re-adapt
#
# PHOME is Home Directory of Adaptor
# DALIB is directory of its implementation for this machine
#
PHOME  = $(HOME)/adaptor
DALIB  = $(PHOME)/dalib/PVM3.1
#
FC  = /vol/lang/f77
NFC = /vol/lang/f77
FSPLIT = /vol/lang/fsplit
CC  = cc
#
# Compiler Options
OPT  = -O -e -w
NOPT = -O -e -w
#
ARCH = SUN4
PVM  = $(HOME)/pvm3
#
XLIB  = $(DALIB)/libadpx.a -lX11
NLIB  = $(DALIB)/dalib.a
LIB1  = $(DALIB)/unilib.a
#
# main programs of host, node, cube
#
MAIN_HOST  = $(DALIB)/mhost.o
MAIN_NODE  = $(DALIB)/mnode.o
MAIN_CUBE  = $(DALIB)/mcube.o
MAIN_NODE1 = $(DALIB)/mnode1.o
#
PVMLIB = $(PVM)/lib/$(ARCH)/libpvm3.a
#
# Object Codes
#
NODE_OBJ = node.o
#
HOST_OBJ = host.o
#
CUBE_OBJ = cube.o
#
OBJ1 = node1.o
#
all: host node
mv node $(PVM)/bin/$(ARCH)
#
host: $(HOST_OBJ) $(MAIN_HOST) $(NLIB) $(PVMLIB)
${FC} -o host $(MAIN_HOST) $(HOST_OBJ) $(NLIB) $(PVMLIB)
#
node: $(NODE_OBJ) $(MAIN_NODE) $(NLIB) $(PVMLIB)
${FC} -o node $(MAIN_NODE) $(NODE_OBJ) $(NLIB) $(PVMLIB)
#
cube: $(CUBE_OBJ) $(MAIN_CUBE) $(NLIB) $(PVMLIB)
${FC} -o cube $(MAIN_CUBE) $(CUBE_OBJ) $(NLIB) $(PVMLIB)
```

```
#
node1: $(MAIN_NODE1) $(OBJ1) $(LIB1)
${FC} -o node1 $(MAIN_NODE1) $(OBJ1) $(LIB1)
#
# So that we don't get the default rules
.SUFFIXES:
.SUFFIXES: .f .o
.f.o:
$(NFC) $(NOPT) -c $<
#
#
# Compiling host sources, node sources
#
host.o: host.f
$(FC) $(OPT) -c host.f
#
node.o: node.f
$(NFC) $(NOPT) -c node.f
#
cube.o: cube.f
$(NFC) $(NOPT) -c cube.f
#
node1.o: node1.f
$(NFC) $(NOPT) -c node1.f
#
clean:
-rm -f *.o node host node1 cube core $(NSRC) $(HSRC) zzz*.f
#
```

Now compare the values of the following variables to the values needed for your machine:

- `FC` is the Fortran compiler for the host program
- `NFC` is the Fortran compiler for the node program
- `OPT` are the options for compiling sources for the host
- `NOPT` are the options for compiling sources for the node
- `ARCH` is the architecture used for PVM (e.g. SUN4)
- `PVM` is the directory where PVM has been installed

If the `Makefile` generated by Adaptor does not work correctly, please fix the necessary changes to get a correct one. These changes should be incorporated in the source to source transformation tool `fadapt`.

Therefore switch to the `src` subdirectory of Adaptor and edit the file `makefiles.c`. Here you can find a subroutine where the makefile is generated for your machine. Make the necessary changes and recompile the sources of `fadapt` by using the $\boxed{\text{make}}$ utility in this directory.

6 Documentation

The following documents exist for Adaptor:

- Installation Guide **iguide.ps** (this document),
- Questionnaire for use of Adaptor **questionnaire.ps**,
- Users Guide **uguide.ps** describes the use of Adaptor,
- Language Reference Manual for Adaptor **language.ps**,
- Description of the Distributed Array Library **dalib.ps**,
- Translation of Data Parallel Programs to Message Passing Programs **trans.ps**

```
cd $PHOME/docs
lpr ... *.ps
```

7 Examples

Many example programs sorted in different directories are given to demonstrate the translation with Adaptor.

```
cd $PHOME/examples
```

Please consider the restriction that one directory can only contain one generated parallel program. This is because always the same file names **host.f** and **node.f**, or **cube.f** or **node1.f** are assigned.

All users that can contribute with new example programs are welcome to send their programs to the author. Suggestions for increasing the performance of the generated parallel programs will be appreciated.

8 Announcing the Installation

After successful installation the following announcement should be made for all users:

```
ADAPTOR has been installed on this machine in the installation
directory $PHOME.

a) Step 1 : Make a link to this directory

   cd
   ln -s $PHOME adaptor

b) Step 2 : Include the bin and man directory in your path

   setenv PATH $HOME/adaptor/bin:$PATH
```

```
   setenv MANPATH $HOME/adaptor/man:$MANPATH

c) You will find examples programs in ~/adaptor/examples.

d) You will find documentation files in ~/adaptor/docs.
   It is recommended to start with the Users Guide (uguide.ps).
```

9 Maintenance and Problems

Users are encouraged to notify the author about the use of the tool and to enter the mailing group for Adaptor (please use the questionnaire in PostScript format in the `docs` subdirectory).

The author would be grateful to all hints for problems and errors that users have had with this tool. Your comments are welcome and will help to improve the quality of this free software product.

References

[Bra93a] T. Brandes. ADAPTOR Language Reference Manual (Version 1.0). Internal Report ADAPTOR-3, GMD, June 1993.

[Bra93b] T. Brandes. ADAPTOR Users Guide (Version 1.0). Internal Report ADAPTOR-2, GMD, June 1993.

[GE90] J. Grosch and H. Emmelmann. A Tool Box for Compiler Construction. *Lecture Notes of Computer Science*, 477:106–116, October 1990.

[NO90] A. Nye and T. O'Reilly. *X Toolkit Intrinsics Programming Manual.* Nutshell Handbooks, Sebastopol, CA, 1990.

[O'R90] T. O'Reilly. *X Toolkit Intrinsics Reference Manual.* Nutshell Handbooks, Sebastopol, CA, 1990.

[Sun90] V. Sunderam. PVM: a Framework for Parallel Distributed Computing. *Concurrency: Practice and Experience*, 3(10), December 1990.