



Scientific Computing & Modelling

Installation Manual

ADF Program System

Release 2004.01

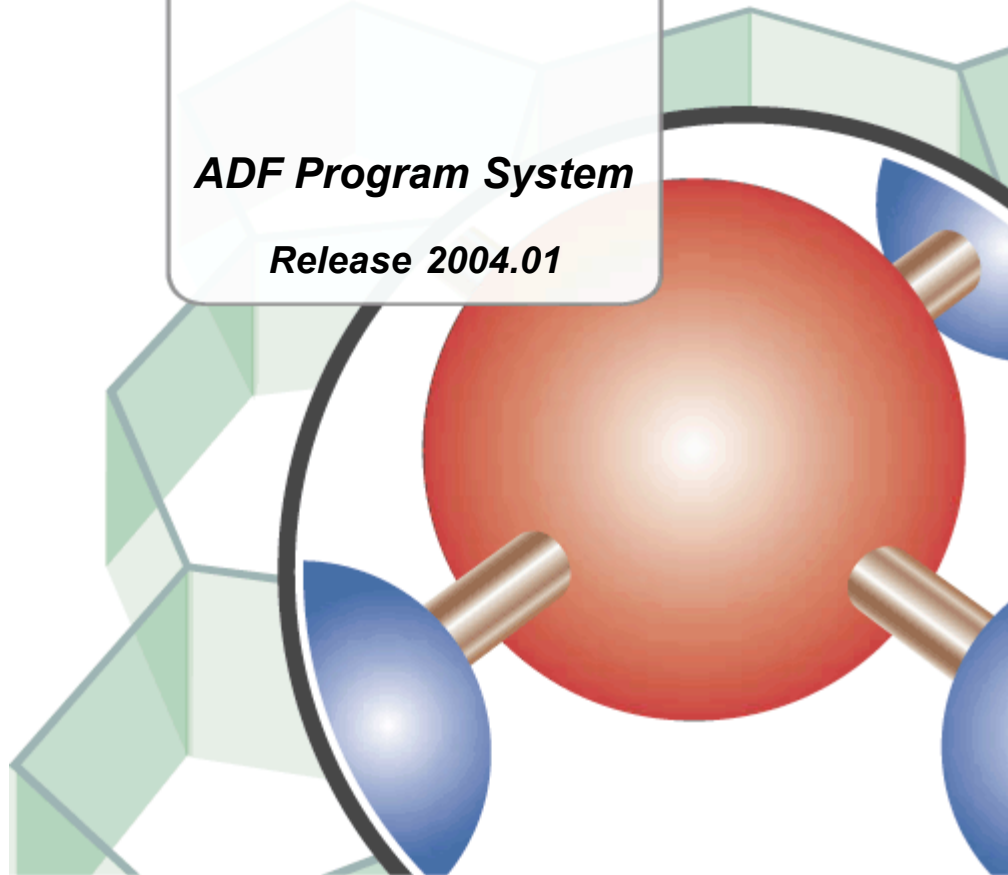


Table of Contents

Installation Manual

1 Introduction

- 1.1 License file
 - License file CORRUPT
- 1.2 Requirements
 - Memory
 - Disk
 - Operating System
 - Network
 - Compiler

2 Installation Instructions

- 2.1 What version to install
- 2.2 External libraries (PVM, MPI, ...)
 - PVM
 - MPI
 - MPICH
 - LAM / MPI
 - For adfplt only: GKS, GR
- 2.3 Download the software
 - 2.3.1 MS Windows
- 2.4 Set up your environment
- 2.5 (Source distribution only): compile the source
- 2.6 Generate license information
- 2.7 Setup the nodeinfo file (PVM only)
- 2.8 Scratch Space (parallel version)
 - Child processes (SCM_TMPDIR, TMPDIR)
 - Master process (SCM_USETMPDIR, SCM_TMPDIR, TMPDIR)
- 2.9 Clean up

3 The ADF package

- 3.1 atomicdata
- 3.2 Examples
- 3.3 source code (adf, band, libtc, libstd, Install, ...)
- 3.4 bin

4 Customizing your Installation

- 4.1 MAXMEMORYUSAGE
- 4.2 SCM_IOBUFFERSIZE
- 4.3 SCM_VECTORLENGTH
- 4.4 SCM_TMPDIR
- 4.5 SCM_USETMPDIR

5 List of Environment Variables

1 Introduction

The Installation Manual describes the installation of the ADF program package on the platforms on which it is supported. For Windows platforms see also the [ADF installation manual for Windows](#). For optimal performance, some system specific tuning may be needed. We therefore strongly recommend to also read Chapter 4 of this document. This Installation Manual is available from the Documentation part of the SCM web site:

<http://www.scm.com/Doc/Welcome.html>

The main programs contained within the ADF package are ADF and BAND. There are also several utility and property programs. These are used primarily for pre- and post-processing data of the main programs. You will always install the complete ADF package at one time. It includes ADF, BAND, and all available utility programs, as well as ADFview and other modules of the ADF-GUI.

The ADF package and support scripts are written for use within a UNIX environment (including Linux). If you want to install it and use it effectively you will need some basic UNIX knowledge. We do **not** provide this general UNIX introduction. If you are new to UNIX we strongly suggest you first make yourself comfortable with this operating system. Very nice tutorials and introductions are available for free on the web, simply search for 'UNIX introduction' or 'UNIX tutorial'.

Typically installation of the ADF package is simple and straightforward. If you have problems installing it, contact us for assistance (support@scm.com).

1.1 License file

The programs of the ADF package require a license file to run. However, you may already install the package without the license file. If this is your first installation of ADF we can, in fact, only make your license file *after* you have installed the package.

The license file contains information about the machines that your license allows you to use the programs on. You must make this file available with read-access for all users of the program on all the machines the program might be used on. You can do this, for instance, by putting the license file on a shared disk, or by copying the file to all machines. The name of the license file is immaterial: you have to set an appropriate environment variable to reference it.

We recommend the following procedure:

1. Install the package as described in Chapter 2.
2. Perform an 'empty' test run, by executing the program with an empty input file:

```
$ADFBIN/adf << eor  
eor
```

This will fail quickly when the program detects that no valid license file exists.

3. The program will print on output the information that we need to create your license file. Send *exactly* that part of the output file to us (email support@scm.com).
4. You will get the license file per e-mail.
5. Pick up the license file and move it to an appropriate location on your machine, corresponding to the environment variable \$SCMLICENSE which you should define in your login script (see Chapter 2).

6. Ensure that the permissions on the license file allow read access for all users.
7. Verify that you can now run the sample runs provided with the package and that they give correct results. We recommend that you consult the Examples document for notes on such comparisons: non-negligible differences are not necessarily indicative of an error.

Note: the sample runs are complete run scripts that should be executed as such, not just input files to feed into the program.

License file CORRUPT

You may find that, after having installed the license file, the program still doesn't run and prints a message "LICENSE CORRUPT". There are a few possible causes. To explain how this error may come about, and how you overcome it, a few words on license files.

Each license file consists of pairs of lines. The first of each pair is text that states in more or less readable format (for a human reader) a couple of typical aspects: A 'feature' that you are allowed to use (for instance 'ADF'), an expiration date, a (maximum) release (version) number of the software and so on. The second line contains the same information in encrypted format: a long string of characters that appear to make little sense. The program reads the license file and checks, with its internal encrypting formulas, that the two lines match. If not, it stops and prints the "LICENSE CORRUPT" message.

So, there are three common reasons why this may happen to you:

1. You are using a license file for another version of the software than your executables correspond to. Newer (major) releases may contain a different encrypting formula, so that the match in old license files is not recognized anymore. In particular, the "LICENSE CORRUPT" error will arise if you run ADF1999 (or later) with a license file that was generated for ADF2.3 (or earlier). Verify that your license file and executable belong to the same major release.
2. The license file as it has been created has been modified in some way. Sometimes, people inspect it and 'clean it up' a little bit, for instance by removing 'redundant spaces', or by making some other 'improvements'. Any such modification will spoil the encryption match and lead to the "LICENSE CORRUPT" error. Sometimes the reason lies in the mailing system: if the encrypted line is rather long, the mailer may cut it into two shorter lines. To verify (and correct) this: edit the license file and see if it really consists of pairs of lines as described above. If not, re-unify the broken lines and try again.

You can use the **fixlic** utility to try to fix this automatically. Please be aware that the **fixlic** utility will try to fix the \$SCMLICENSE file, and replace it with the fixed copy. Thus, you need to make a backup of your license file first, and you need to have write access to the license file

```
cp $SCMLICENSE $SCMLICENSE.backup
$ADFBIN/fixlic
```

3. On some machines (and again, the mailer may play a role here as well) the different lines are separated by two (invisible) characters: <CR> and <LF> (carriage-return and new-line), while on others (and ours) there's only a <LF> to separate the lines. The extra <CR> leads to the "LICENSE CORRUPT" error.

We have included a small utility in our package (fix_license, located in the \$ADFBIN directory after installation) to correct this problem. Type (assuming the SCMLICENSE environment variable has been set correctly):

```
fix_license $SCMLICENSE
```

and any <CR> characters will be removed from license_file. (Your LANG environment variable should preferably be set to 'C').

1.2 Requirements

The requirements for the ADF package vary from platform to platform. A list of supported platforms, with information on the operating system (versions), parallel environment (MPI, PVM, ...), and compilers used is available on the Download section of our web site:

<http://www.scm.com/Downloads/Welcome.html>

Memory

The amount of memory you need depends greatly on the kind of calculation you perform. For most calculations 256 MB per CPU will be sufficient, but if you have lots of memory available most program will run significantly faster. It also reduces the amount of IO, which may speed up your calculation depending on your IO system and your operating system.

Disk

For the installation of the package you need from less than 256 MB (without sources) to 512 MB (with sources, and compiled objects). To run the programs, it greatly depends on what type of calculation you perform. For most ADF calculations 1 GB of disk space will be enough. For BAND calculations you will need a few GB's up to a few tens of GB's (for huge calculations) of free disk space, depending on your calculation.

Operating System

The package will run with most recent UNIX operating systems, including Linux.

A Windows version is also available. It was tested on the most popular Windows varieties, Windows 2000 and Windows XP. See the [ADF installation manual for Windows](#) for more detail.

For Macintosh OS X you will need at least 10.3 (Panther), and you need to install the X11 environment as well (it is available as an optional install on the Panther installation disks).

If your UNIX platform is not listed, please contact us for more information.

Network

The serial version of the software needs no special network facilities. The parallel version needs some kind of network for communication between nodes if you are running on more than one node. If you are not using more than eight nodes, a switched Ethernet network is typically sufficient for good parallel performance. If you are using more nodes you will need a switched FastEthernet connection, or some other fast communication method, such as Myrinet.

All commercial parallel machines (SGI, IBM, Compaq, HP, ...) have very good communication facilities, so the package should run fine on those platforms as far as the communication system is concerned.

Compiler

If you have a license for the source code, you can compile and link the source yourself, with or without your

own modifications.

The source consists mainly of Fortran90 code, with some small parts written in C. You need to use the same Fortran90 compiler as we are using on your platform, since some parts of the code are available only as object modules. It is very unlikely that other compilers will work with these object modules. We cannot support other compilers than we are using ourselves.

To check what compiler to use, check the detailed machine information on the Download section of our web site.

2 Installation Instructions

To install the ADF package you have to go through the following steps:

- 1. what version to install (serial, PVM, MPI, ...)
- 2. external libraries (PVM, MPI, ...)
- 3. download and unpack the software
- 4. set up your environment
- 5 (Source distribution only): compile the source
- 6. generate license information
- 7. setup the nodeinfo file (PVM only)
- 8. scratch space (parallel version)
- 9. clean up

2.1 What version to install

If you are installing the package on a machine that has only one CPU you should install the **serial version**. For some platforms we have no serial version available. In such cases you can just install the parallel PVM version.

For Windows, there is only serial version available.

If you are installing the package on a machine with more than one CPU, either on the same board or on some distributed machine (for example a Linux cluster of Pentium machines), you should install the **parallel version**.

The parallel version does not require you to run in parallel: you can use it to run in serial mode as well.

2.2 External libraries (PVM, MPI, ...)

PVM

PVM (Parallel Virtual Machine) provides a common interface for a large number of different parallel architectures, ranging from a heterogeneous cluster of workstations to shared memory multiprocessors. It is public software, and you can download it for free. Some vendors have their own optimized implementations of PVM to exploit specific hardware features. Unfortunately, these special versions are not always flawless.

The PVM version of the ADF package, when available, always uses the public version of PVM, and on most platforms it uses the default configuration of PVM.

You will need to know what version of PVM has been used to generate the executables. Usually, correct execution cannot be guaranteed if a different PVM version is used. That information on our web site: <http://www.scm.com>

If you are not familiar with PVM and need to install it yourself, we strongly suggest you have a look at the PVM documentation. PVM (including FAQ and Users'Guide and Tutorial) can be downloaded from:

<http://www.netlib.org/pvm3/index.html>

To use PVM you need to:

- install PVM (see PVM documentation)
- set two environment variables: PVM_ROOT and PVM_ARCH
- add \$PVM_ROOT/bin/\$PVM_ARCH and \$PVM_ROOT/lib to your PATH
- make sure pvm can start the pvmd on all nodes of your system

(you may have to look at .rhosts or ssh issues)

To run in parallel you start the programs as usual, but you (or your batch system, job script, ...) need to start and stop the PVM virtual machine yourself. It is convenient to make a special 'submit' script that handles this for you if you are using a batch system.

- make submit script to start and stop PVM

Important: you should use

- PVM_ARCH=**SGI6** on all SGI platforms, and
- PVM_ARCH=**MACOSX** on the Macintosh OS X platform.

If you choose something different you will certainly encounter difficulties.

We strongly advise to use the same version of PVM as we have used to generate and test the ADF package.

MPI

MPI (Message Passing Interface) is a standard describing how to pass messages between programs running on the same or different machines.

The functionality provided by MPI implementations is very much like PVM. However, since MPI is a formal standard it is being actively supported by all major vendors, PVM is not. Some vendors have highly-optimized MPI libraries available on their systems. There are also a couple of public implementations of the MPI standard, such as MPICH and LAM / MPI. The ADF package supports a number of vendor versions of MPI (for SGI, HP, IBM), and MPICH and LAM / MPI.

If you wish to use the MPI version of the ADF package you should determine if one of the standard versions of MPI is already available for you, or if you need to install it yourself.

MPICH

MPICH is a portable implementation of the MPI standard. You can get both the software and lots of documentation from the MPICH web site:

<http://www-unix.mcs.anl.gov/mpi/mpich/>

ADF uses the ch_p4 device.

- Install MPICH following the MPICH documentation.

For compiling, linking, and running, ADF needs to know where your MPICH ch_p4 installation lives. You need to set the MPIDIR environment variable to this value, and you need to add \$MPIDIR/bin to your PATH.

- Set MPIDIR, and add \$MPIDIR/bin to your PATH

Example

```
MPIDIR= /usr/local/mpich/1.2.4/ch_p4
```

```
export MPIDIR
PATH=$MPIDIR/bin:$PATH
```

Furthermore, at runtime you need to specify what machines to run on. You can do this by making a file that should contain the names of all nodes (one per line) you will use in your parallel job. ADF needs to know what name you use for this file. You pass this information to ADF by setting the environment variable `SCM_MACHINEFILE` to the absolute path of your machine file.

If you are using a batch system on your machine, it would be convenient to make a submit script that automatically creates the correct machine file for you.

→ Make script to build machine file, and set `SCM_MACHINEFILE`

Example (for use within the PBS batch system):

```
SCM_MACHINEFILE=$HOME/.machines.$$
export SCM_MACHINEFILE
uniq < $PBS_NODEFILE > $SCM_MACHINEFILE
```

Note about Linux and MPICH. The current MPI version of ADF for Linux is linked with MPICH 1.2.5.2 which was compiled using gcc 3.2.2 and Intel Fortran 8.0. You may need to install the same version of MPICH compiled with Intel Fortran compiler and not g77.

LAM / MPI

LAM (Local Area Multicomputer) / MPI is also an implementation of the MPI standard. You can get both the software and documentation from the LAM / MPI website:

<http://www.lam-mpi.org>

→ Install LAM / MPI following the instructions from the documentation.
→ Make sure `$HOME/lammpi/bin` is in your `PATH` (assuming you installed LAM/MPI in your home directory).

When you want to run a parallel job you need to initialize the LAM/MPI system first. You do this with the **lamboot** command. It needs a hostfile just like PVM and MPICH. When your calculation is finished, you might want to clean up all LAM/MPI processes by using **lamhalt**. Assuming you are running a batch system, we advise that you put the **lamboot** and **lamhalt** commands in a submit script so it will be handled automatically.

The LAM/MPI version of ADF uses `mpirun` to start the programs. Thus `mpirun` should be in your path (which `mpirun` should point to the LAM/MPI version of `mpirun`). When the program finishes, it will also clean up (but leave the LAM/MPI environment intact) using the `lamclean` command.

For adfplt only: GKS, GR

The graphical libraries GKS and GR are required to install and operate the *adfplt* utility program in the ADF package, for graphical display of orbitals, densities and potentials. It is available for free from the authors, at

<http://www.KFA-Juelich.de/zam/CompServ/software/graph/soft.html>.

You need the GR and GKS libraries. For GKS: go to GLI and select GLI/GKS. These libraries are available (or working correctly) only for a few platforms, unfortunately. The ADFview program provides an alternative that is available on all currently supported platforms.

2.3 Download the software

If you have not already done so, you need to download the software from the Download section of the SCM web site using a web browser. You will be asked for a username and password. You should have received these from SCM (contact admin@scm.com if you do not have a username and password yet).

If you want to install the precompiled executables, you need to download **adf2004.01.bin.tgz** for your platform. If you have a source code license and want to compile the source code yourself you also need to download **adf2004.01.src.tgz**.

→ Download the **adf2004.01.bin.tgz** and, if applicable, **adf2004.01.src.tgz** for your platform (from www.scm.com).

(use browser to download files)

If you want to install the precompiled executables only (thus skip this step if you have a source license and will compile adf yourself): gunzip and untar the downloaded files:

```
gunzip adf2004.01.bin.tgz
tar -xf adf2004.01.bin.tar
```

A new directory **adf2004.01** will be created containing your new adf installation.

2.3.1 MS Windows

For Windows, you need to download **adf2004.01.setup.exe** and run it. It will install everything and update the environment variables accordingly. On Windows 2000/Windows XP you must to be a member of the Administrators group to perform the installation. On Windows 98 you will need to reboot your computer to finish the installation.

2.4 Set up your environment

You will need to define a couple of environment variables, preferably in your *login* script.

If you are using some kind of batch system, make sure that they are set not only for interactive work, but also for batch use. In a distributed parallel environment, like a Linux cluster, make sure that these environment variables are set on all nodes.

All these environment variables are also automatically passed on from the master process to the child processes in case of a parallel run. Thus the values set on the node where you start the program will override any value you may have set on the nodes where the child processes are running. Typically, you should still set the environment variables on all nodes since you won't know in advance on which node the master will run.

Environment variables: ALL versions

variable	value	comments
ADFHOME	\$HOME/adf2004.01	(full path) main directory (see 2.3)
ADFBIN	\$ADFHOME/bin	(full path) directory for scripts and binaries
ADFRESOURCES	\$ADFHOME/atomicdata	(full path) directory with the ADF database (basis sets and so on)
SCMLICENSE	\$ADFHOME/license	(full path) the (future) location of your license file

Environment variables: PARALLEL versions

variable	value	comments
NSCM	16	(number) default number of parallel processes
SCM_TMPDIR	/scratch/\$USER	(full path) directory where child processes should create scratch files
SCM_USETMPDIR	yes	master process should also use SCM_TMPDIR

Environment variables: PVM version

variable	value	comments
PVM_ROOT	\$HOME/pvm3	(full path) directory with PVM installation
PVM_ARCH	SGI6	architecture code for your machine (NOTE: use SGI6 on SGI)

Environment variables: MPICH version

variable	value	comments
MPIDIR	/mpich/1.2.4/ch_p4	(full path) directory with ch_p4 MPICH installation
SCM_MACHINEFILE	\$HOME/.machine	(full path, MPICH only) file listing nodes to use

Environment variables: only for ADFPLT

variable	value	comments
GKSGT_LIB	/usr/local/lib	(full path) directory with GKSGR lib

2.5 (Source distribution only): compile the source

The downloaded source should be unpacked using gunzip and tar:

```
gunzip adf2004.01.src.tgz
tar -xf adf2004.01.tar
```

The result will be a new adf2004.01 directory containing the sources.

After unpacking and setting your environment properly, you should run the configure script. This script is located in the \$ADFHOMe/Install directory, and should be executed from the ADFHOMe directory. It takes care of creating a bin directory, populating it with a couple of required scripts, and creating the Makeflags and settings files you need to compile the ADF sources.

→Run Install/configure

Example:

```
cd $ADFHOMe
Install/configure
... answer the questions asked by the configure script ...
```

Next **unpack the binary distribution**. This installs a couple of programs in the \$ADFBIN directory that are not included in the source distribution.

```
gunzip adf2004.01.bin.tgz
```

```
tar -xf adf2004.01.bin.tar
```

Next, you can compile and link the ADF sources by executing the yam (Yet Another Make) script. yam is installed in the \$ADFBIN directory by the configure script.

```
cd $ADFHOM
$ADFBIN/yam
```

After a while, depending on the speed of your computer, ADF should have been compiled and ready to use, just as if you had installed the precompiled executables.

2.6 Generate license information

Most of the programs that form the ADF package require a license file to run. This file contains information about your machine, the version(s) of the package for which you have a license, and about the end date of your license.

You will receive the license file from SCM, but SCM first needs to get some information from you about your machines. ADF program generates this information by running on your machine when no (valid) license file is available.

To make sure that ADF does not find a valid license file, you may need to set the environment variable SCMLICENSE to point to a nonexisting file **temporarily**.

Note that SCMLICENSE refers to a name of a file, it is not a name of a directory.

- Run ADF with an empty input file, or run ADF interactively and terminate it with control-D.
- Mail the resulting output to support@scm.com

SCM will prepare a license file for you, according to your license conditions, and mail it to you with instructions on how to install it.

Example:

```
cd $ADFHOM
SCM_LICENSE=$HOME/jojohojojo
export SCM_LICENSE
$ADFBIN/adf << eor
eor
```

This will produce the output very similar to the following:

```
=====
```

```
You need to install a valid license file.
Checked: /home/visser/jojohojojo
If you do not have a license file, contact SCM
(E-mail:support@scm.com) with the following
information:
```

```
release: 2004.01
:chem.vu.nl:
:Darwinmolybdenu00:03:93:4c:37:a8:
```

For more information, please consult the
INSTALLATION MANUAL

```
=====
```

STOP LICENCE FILE NOT FOUND

(... and mail this output to support@scm.com)

Please refer to [ADF for Windows release notes](#) on how to obtain the license information on Windows.

2.7 Setup the nodeinfo file (PVM only)

If you have installed the parallel PVM version of the ADF package you may need to set up the nodeinfo file.

The nodeinfo file tells ADF how many tasks are allowed to be run on each node in the PVM virtual parallel machine.

If you have a cluster of single CPU workstations you most probably want to generate at most one task per node. Since this is the default, you do not need to do anything special.

If one or more of your nodes have more than one CPU, you probably want to generate also more than one task on those nodes. Normally you should generate one task per CPU. The nodeinfo file communicates that information to ADF.

The format of the nodeinfo file is:

- one line per node,
- for each line: node name and a number, separated by a space.

Here node name is the name of your node, as shown using the `pvm conf` command. Number is the *maximum* number of tasks you wish to use on that node. Nodes which are not specified will default to one task to be generated at most.

Thus, if you have two SGI workstations (`silly1` and `silly2`) with 4 CPU's each, the contents of your nodeinfo file should be (two lines):

```
silly1 4  
silly2 4
```

You should put this nodeinfo file in `$ADFBIN` and make it readable for all ADF users.

For some special situations it makes sense to have a unique nodeinfo file for each calculation (for example when using a batch system with a large machine). For this purpose the programs within the ADF package first look for a local nodeinfo file (in the directory in which the program is started), and next it will try to use the `$ADFBIN/nodeinfo` file. If both do not exist, it will assume that it may generate one task per node.

2.8 Scratch Space (parallel version)

You may skip this section if you have installed the serial version of the ADF package.

All processes (the main process started by you and the child processes) will write to disk, often a significant amount of data. You will have to make sure that the scratch space used by all these tasks is both big and fast enough. Use the environment variables `SCM_TMPDIR` and `SCM_USETMPDIR` to tell the ADF programs where to create their scratch directories.

In the next few sections we will explain exactly what these variables do.

→ Set the `SCM_TMPDIR` and `SCM_USETMPDIR` environment variables.

Example:

```
SCM_TMPDIR=/local/scratch/$USER
export SCM_TMPDIR
SCM_USETMPDIR=yes
export USETMPDIR
```

Child processes (SCM_TMPDIR, TMPDIR)

The child processes will create a scratch directory. For efficiency reasons, that directory should reside on a local disk if possible. You need write access to that directory, and it should be big enough.

The child processes will create their scratch directories within **\$SCM_TMPDIR**. This environment variable **will be passed** from the main program (master) to the child processes! Thus, you may set it in your run script.

If SCM_TMPDIR has not been set, the child processes will create their scratch directories in **\$TMPDIR**. This environment variable **will NOT be passed** from the main (master) process to the child processes! Often it is set correctly by the batch system.

Master process (SCM_USETMPDIR, SCM_TMPDIR, TMPDIR)

The master process will by default write to the directory **in which it was started**.

You can change this by setting the **SCM_USETMPDIR** environment variable to 'yes'. In that case, the master process will create a scratch directory just as the child processes do. Typically this is a good idea for performance reasons.

The master program will take care of copying required files from the startup directory to the scratch directory, and it will also copy the result files back to the startup directory. Thus you can use it as if it is running in the directory in which you start it.

2.9 Clean up

At this point you may clean up the \$ADFHOMEDIR directory a little.

You may remove the *.tar files if you have not already done so, and if you are short on disk space you may remove the examples directories. We do advise, however, to keep them around since they contain nice examples that may be used to get acquainted with ADF.

3 The ADF package

3.1 atomicdata

The directory `atomicdata/` contains the database. You need it when you start to do calculations. The database consists mainly of basis sets for all atoms, each with its own advantages and disadvantages (regarding accuracy and size for example).

The basis sets are described in detail in the ADF BasisSet document, available from the Documentation part of the SCM web site (<http://www.scm.com>).

3.2 Examples

The directory `examples/` contains sample run scripts and outputs. The run scripts are shell scripts. The input is provided within the script. Thus these scripts should be executed, and not given as input to ADF.

Just look inside the scripts to see what input is actually given to the program.

The example calculations are extensively documented in the Examples document, available from the SCM web site (<http://www.scm.com>).

3.3 source code (adf, band, libtc, libstd, Install, ...)

The source code files, which can be found in the program and library directories and subdirectories thereof, need to be pre-processed by the parser (`scu`) before compilation and are denoted by suffices `".d"`, `".d90"`, `"*.dmod0[0-9]"`, `".cd"`. The parser produces files containing plain FORTAN or C code having a suffix `'.f'` or `'.c'`. Other source files (used by the parser) are include files (files with extension `".h"`) and files which define names (the file `"settings"`

Parsing and compilation are controlled by `$ADFBIN/yam`.

The directory `Install/` contains a script `configure`, some data files which provide generic input for `configure` (`start`, `starttcl`, and some more), a portable preprocessor `cpp` (based on Mouse `cpp`), the `adf` preprocessor (`adfparser`) and files that will be addressed by the `$ADFBIN/yam` command.

Apart from a few more files that are of little importance in the installation, you'll find in `Install/` a number of subdirectories with names like `pentium_linux`, `sgi_irix65`, and so on: they refer to the platforms that are supported.

`Install/configure` is an interactive script. It will ask some data from the user: what platform, what version (PVM, MPI, serial), and how many CPU's to use at most.

The `configure` script will create the files `settings` and `Makeflags` in `$ADFHOM/`, based on the files in the `Install` directory. It also creates a few 'Makefiles' in `$ADFHOM/` and in subdirectories thereof. It creates the `$ADFBIN` directory.

3.4 bin

When the programs have been installed, the binary executables have been placed in `$ADFBIN`, typically `$ADFHOM/bin`, with names like `'adf.exe'`, `'band.exe'`, and so on. There will also be files `'adf'`, `'band'`. The

latter are scripts that execute the binaries.

You should use the script files, rather than the executables directly. The script files take care of several convenient features (like the BASIS key described in the ADF User's Guide), and they provide a correct environment for running in parallel. See also the sample run scripts and the Examples document.

4 Customizing your Installation

You can customize and optimize your ADF package by changing or setting several environment variables, and/or use some input options. We advise you to test the effects of changing them for some of your typical calculations, and then setting the optimal values in the scripts that define the environment for your login shell. Don't forget to make sure that the changes you make will also apply for batch jobs, and for all nodes of a distributed computing system.

A complete list of environment variables used by the ADF package can be found at the end of this document. Here we will discuss a few which are particularly important.

4.1 MAXMEMORYUSAGE

This option is currently only available in your input file.

You can set the maximum amount of memory (in MB) available to some parts of the code. If you run in memory problems, or your block lengths are very small, you may play around with this input option. Both increasing and decreasing might help and have influence on the performance, just try for some of your typical calculations. Typically you do not need to modify MAXMEMORYUSAGE if your vectorlength is still equal to its default and your calculation proceeds normally.

4.2 SCM_IOBUFFERSIZE

Most programs within the ADF package use the KF IO system. This is coupled to a facility to store (parts of) files in memory, if you have enough memory available. Thus, the programs will cache the IO data instead of the operating system, and reduce the amount of IO significantly.

It depends on your operating system and hardware if you can benefit from this scheme. The default is a small buffer (8 MB), which has only a small performance benefit. Sometimes this is all you can get, but in some cases you can have a major performance improvement by making this buffer much larger, for example 128 MB or 256 MB. You do this by setting the SCM_IOBUFFERSIZE environment variable.

Please try for yourself, with your typical calculation on your production machine to find out the optimal value for your situation.

4.3 SCM_VECTORLENGTH

Almost all programs within the ADF package use numerical integration, and this is normally the most time-consuming part of the code. Numerical integration involves summing together results for each 'integration point'. By handling a number of points together you can greatly influence the performance. The number of integration points handled together is called the block length.

If the block length is too small, you will have a significant overhead and the programs will be very slow.

If the block length is too large, lots of data will not fit in cache memory and again the program is not at optimum speed.

The correct block length is somewhere in between, ranging from 32 to 4096 depending on your hardware. Sometimes it pays off to set the block length explicitly NOT to a power of 2 to avoid memory bank conflicts.

Again, try for yourself, with your typical calculation on your production machine to find out the optimal value

for your situation. On most machines, 128 is a good compromise value.

4.4 SCM_TMPDIR

This environment variable determines where the child process, and possibly the master process, will create their scratch directories. Setting this variable correctly is extremely important for performance reasons. For an extensive discussion see 2.10.

4.5 SCM_USETMPDIR

Set this variable to force the master process in a parallel calculation to use SCM_TMPDIR as its scratch directory, just like the child processes. For more information, see 2.10.

5 List of Environment Variables

List of environment variables

Environment variable	Typical value	Description
NSCM	8	Number of parallel processes to use
SCM_IOBUFFERSIZE	32	Size of memory buffer (MB) for KF files
SCM_VECTORLENGTH	128	Block length used
SCM_TMPDIR	/scratch/\$USER	path to fast scratch directory (2.10)
TMPDIR	/tmp	path to fast scratch directory (2.10)
SCM_USETMPDIR	yes	Master also uses SCM_TMPDIR (2.10)
SCM_RCV_TIMEOUT	900	(PVM only) time (seconds) to wait for messages
SCM_MAX_RCV_TIMEOUTS	2	(PVM only) maximum number of timeouts
SCM_ALARMTIME	1800	time (seconds) to wait for messages using ALARM signal, 0 disables timeouts [currently the default setting is zero, because this feature may not work on some platforms]
SCMBCOP	1	broadcast algorithm
SCMGAOP	5	gather algorithm
SCMCBOP	1	combine alogrithm
SCMSPAWNSCRIPT	adfs	script to start child processes
ADFBIN	\$ADFHOMe/bin	directory with executables
SCM_DEBUG		set to debug SCM_TMPDIR
TEMPDIR		old name for SCM_TMPDIR
SCM_NOMEMCHECK		disable memory checks