This file gives a quick overview of how to compile the LINSOL package
on your machine.

Prerequisites: you need the GNU make utility, a C compiler, a FORTRAN90
compiler (and the 'tar' utility) to install LINSOL!

LINSOL is 'autoconfiscated', this means that you should get a
(non-optimal) working version with the commands

```
> mkdir LINSOL
> cd LINSOL
# Download of the Source code of LINSOL into the actual directory!
tar -zxvf LINSOL<version>.tar.gz
> ./configure  # see chapter 'Error handling -1-', if errors occur
# You only have to do the folllowing, if you install LINSOL on a vectorcomputer!
# Adapt the Include-files 'include/norec.h' and 'include/scalar.h' to
# your vectorcomputer. Don't do this, if you don't know the right compiler
# options!
> make         # see chapter 'Error handling -2-', if errors occur
> make test    # see chapter 'Error handling -3-', if errors occur
```

If all goes well you'll end up having an executable >linsol in the
'LINSOL/bin' directory and two libraries 'liblinsol.a' and 'libnocomm.a'
in the 'LINSOL/lib' directory. This however is only a serial version with
probably non-optimal cache support. Usually you have to type >make
a second time to link the executable >linsol!!! In the automatic
test three independent runs of the LINSOL program package are performed.
The first test has run well if the solution of it's run is (1,2,3,4,5,6,7,8)
with rounding errors. In the following two test runs the realtive norms of
the errors should be below 10**(-6). Then the test runs are o.k.

Also note that you might need the GNU make utility if your version of
make fails immediately.

In order to compile an optimal LINSOL for your system you have to give
the appropriate options to the >configure script as described in chapter
"Passing options to configure".

If you are a system administrator and provide the libraries, the executables
and the shell scripts to the users of your computer site, you should
move the libraries to a directory with public library entries, the binaries
to a directory with public binaries and the shell scripts to a directory with
an entry in the PATH environment variable.

Only for installations with batch system:
Via the shell script >linnox batch requests can be launched.
The files 'LINSOL/bin/linsh', 'LINSOL/bin/shscript_template' and
'LINSOL/bin/shscript_template1' are prepared for the useage of the batch
system NQS. If you don't use NQS, you must adapt the above mentioned files
to your computer site. The following notes are only valid for systems
supporting NQS:
the shell script 'LINSOL/bin/shscript_template1' launching serial batch jobs
must eventually be adapted to your computer site. Check the NQS commands in
the file 'LINSOL/bin/shscript_template1'!
the shell script 'LINSOL/bin/shscript_template' launching parallel batch jobs
must be adapted to your computer site. Check the NQS commands in
the file 'LINSOL/bin/shscript_template1' and adapt the command line launching
parallel jobs to your computer site!

Only for installations on parallel computers:
Via the shell script >linnox interactive parallel jobs can be launched.
If you want to do this, the command line launching interactive parallel jobs
must be adapted to your parallel computer site in the file 'LINSOL/bin/linsh'.

Only for installations on Cray parallel computers:
the installation step for parallel computers
(see also "Passing options to configure")

```
> ./configure  --with-maxnpe=<p>
```

must be replaced on Cray parallel computers by

```
> LINSOL_PLIB=MPICRY ./configure  --with-maxnpe=<p>
```

```
Error handling
--------------

-1-

The first step that could go wrong on your site is the configuration
of the system-dependent variables and the creation of the Makefiles
by calling

> ./configure ...

 - if the configure-script can not detect the type of the host (that is
   done automatically be the shellscript config.guess), probably the
   script config.sub is too old! Please substitute the script config.sub
   (and eventually config.guess) by the latest version(s). You will find
   both scripts in http://ftp.gnu.org/gnu/autoconf/autoconf-2.sv.tar.gz
   (autoconf-2.sv.tar.gz is presently (August 2004) autoconf-2.59.tar.gz).

 - if the configuration of the system-dependent variables fails, please
   adapt the variables of the shellscript 'do_config' to the data of your
   computer site and call

   > do_config

   In the direcoty 'LINSOL/config_scripts' you find shellscripts for
   the configuration of some installation sites. You can use them as
   templates if you want.

 - Within the configuration step the data types are also checked. If this
   check does not work, i.e. the Fortran data types real*8, integer and logical
   have zero bytes, please go into the directory './source/linsol' and set
   the variables ireal, iint, and ilog in the file 'constants.f'.
   Usually you have to set ireal=8 (i.e. real*8 is 8 bytes long), iint=4 and
   ilog=4 (i.e. integer and logical are 4 bytes long).

Hopefully it will work then.


Remark for Linux users:

   If you are using Linux, you can set your Linux flag settings beneath
   *-linux*) in the file configure.in. Please call then

   > autoconf
   > ./configure ...

   The safest way to configure LINSOL for Linux is to adapt the shellscript
   'LINSOL/config_scripts/do_config_linux_nag_f95' to your installation
   site, to copy it into the file 'LINSOL/do_config' and to call

   > do_config


-2-

If >make fails, try >gmake! If the compilation of the routines starts,
compiler errors may occur. Here I can only give general hints what to do then.

 - If any modules are not found during the compilation process, please check
   if the chosen compiler uses '.mod' as suffix for modules. If not, please
   set the environment variable  MOD to the suffix used by your compiler.
   In the shellscript 'module_mover' you will find the standard transaction
   for modules. If the standard transaction is not sufficient, please
   adapt the shellscript to your realities.
 - Reduce the optimization level and try again.
 - Check your compiler options! Don't use autodouble functions for any basic
   data types!
 - LINSOL has a few C routines. Add -DFTNLINKSUFFIX to your compiler options,
   if an underline character following the name of the C routine is required
   for calls from FORTRAN. Add -DUPPERCASE to your compiler options, if the
   names of the C routines must be uppercase on your computer site!
 - Remark for Linux users: if you are using the NAG Fortran95 compiler, you
   must set the compiler option '-dusty' to avoid errors during the compilation
   of the program package LINSOL.
 - If there are any other compiler errors, check the source code.
```

```
-3-

If
> make test
fails, try
> gmake test
If the first test fails, the link step for examples <exam01> will
probably fail. Please check the pathes to the libraries. If the
shell script in the second and third test run does'nt work,
set the shell used by the shell script 'LINSOL/bin/linnox' to the kshell,
i.e. replace !#/bin/sh by !#/bin/ksh in the files 'LINSOL/bin/linnox'
and 'LINSOL/bin/linsh'.



Passing options to configure
----------------------------

You can get a summary of all available options with

> ./configure --help

Apart from the standard options that you can pass to a 'configure'
script, the LINSOL 'configure' understands two additional switches:

--with-maxnpe=<p>

Specifying this option will configure the LINSOL package for parallel
use with at most <p> processors. In order for this to work you'll need
a parallel programming library supported by LINSOL. The standard message
passing system supported by LINSOL is MPI. The script 'configure' will
check for this of course and alarm you if it cannot find it.

--with-cachesz=<w>

This option will configure the LINSOL package to make optimal cache
use for a cache size of <w> double precision words (i.e. 8 Byte
words).


Overriding 'configure' decisions
--------------------------------

You can easily override most of the flags or values that 'configure'
will built into the Makefiles. In order to do so just specify the
variable name before you call 'configure'. E.g. to use the native C
compiler 'cc' instead of the GNU C compiler you could type

> CC=cc ./configure

In this way you can override the variables

F77          - Fortran77 compiler
F90          - Fortran90 compiler
CC           - C compiler
FFLAGS       - Fortran compiler flags maximal optimization
FFLAG2       - Fortran compiler flags without full optimization
               (for some critical files)
FDBFLAGS     - Fortran compiler flags for debugging
FLDFLAGS     - Linker flags for generating executables
CFLAGS       - Flags for the C compiler
F77MP        - Special command to compile and link a Fortran77 code with
               the MPI library.
F90MP        - Same for Fortran90 code.
LIBFLG       - to include the MPI library into all link steps for the creation
               of parallelized running executables; you don't need this flag,
               if there are special commands to compile and link Fortran code
               with the MPI library (see F77MP, F90MP).
LINSOL_PLIB  - Name of the message passing library. Usually it will
               be chosen automatically. On Cray parallel computers it
```

```
                must be set
                >LINSOL_PLIB=MPICRY ./configure ...



Example
-------

Configuring LINSOL as a parallel package on at most 256 processors
with 64KB cache using the native `cc' C compiler:

> CC=cc ./configure --with-maxnpe=256 --with-cachesz=8192

Compiling afterwards with >make or >gmake will produce the same libraries
and the same binary `LINSOL/bin/linsol' as if you would install LINSOL without
the option `--with-maxnpe' and additionally the library libMPI.a and the
executable `LINSOL/bin/plinsol' for parallel runs.



Debugging (is not supported in the standard version)
---------

It is easily possible to create a version of LINSOL that can be debugged.
All what you have to do is to set the environment variable DEBUG by
the command
> export DEBUG=true
before you are starting the compilation process by the command
> make  (gmake)



Multiple architecture builds
----------------------------

When binaries for multiple architectures are required then the
`autoconfig' setup makes it possible to compile them from a single
source tree if you follow the standard procedure for this case:

1) Create a subdirectory for the build-architecture just below the
   LINSOL top directory, for example if you build below your home
   directory:

   > cd ~/LINSOL
   > mkdir aix-sp2

2) Change into the newly created directory and configure the package
   as you normally would only by using `configure' from the parent
   directory:

   > ../configure <usual-options>

3) Compile as you normally would, e.g.

   > make

This will leave you with only the architecture-dependant files in the
directroy tree below the newly created architecture directory. All other
files reside in the main tree as expected.

Just in case you wondered why there are two include path switches to
the compiler - this is to support the situation described above. The
architecture independant files reside in the "normal" include
directory while the `configure' generated files are located in the
subtree.



Cross-compiling
---------------

Warning: This is untested.

You should be able to cross compile LINSOL if your build environment
supports cross compilations. Just specify `--target' and `--build',
`--host' if needed as parameters to `configure'.
```

```
You should also check the parameter files that are generated as some
information needed in there CAN NOT be determined in a cross compiling
environment, as no programs can be run on the target machine during
the configuration phase.

If you suceeded in cross compiling for a new platform then please let
us know! If you did not succeed then let us also know - we should be
able to find a way to make it work.
```