# Installation Notes
# Southampton *High Performance Fortran* Translation System, Beta release 0.1

D.B. Carpenter

June 8, 1995

## 1 Prerequisites for installation

The translation system works by translating HPF to standard Fortran 90, with calls to an HPF run-time system written in C++; the run-time library is layered on MPI. Besides the standard UNIX utilities, prerequisites for installation are

- A standard Fortran 90 compiler.
- A C++ compiler.
- An MPI installation.

The combination of these three systems is the *software platform* for the HPF installation. Section 5 records the platforms on which we have successfully installed HPF so far. If you have a significantly different platform, please contact us at `info-hpf@par.soton.ac.uk` or `bug-hpf@par.soton.ac.uk`. We will make our best efforts to help with the installation (and incorporate the knowledge gained into subsequent releases).

Public domain releases of C++ are available from

- The GNU compiler is available in the UK from the FTP site at imperial college: `ftp://src.doc.ac.uk/pub/GNU/gcc-2.6.3.tar.gz`. You will also need `ftp://src.doc.ac.uk/pub/GNU/libg++-2.6.2.tar.gz`.

Public domain releases of MPI are available from

- The Argonne National Laboratory implementation, MPICH, is available from `ftp://info.mcs.anl.gov/pub/mpi`. For further information, see the web page at `http://www.mcs.anl.gov/mpi/index.html`.

## 2 Distribution

The HPF distribution is contained in the archive file `shpf.0.1.tar`. The beta release is available by FTP from `par.soton.ac.uk` in the directory `/pcsg/hpf_trans`. Contact us at `info-hpf@par.soton.ac.uk` to receive the password.

Working in a suitable root directory, extract the archive file. For example

```
tar xvf shpf.0.1.tar
```

This creates a directory called `shpf` in the current directory. `shpf` is now the home directory for the HPF installation.

## 3 Configuring and building HPF

1. Ensure the C++ compiler and the Fortran 90 compiler are somewhere on your path.

2. Change directory to the HPF home directory, eg

   ```
   cd shpf
   ```

3. Configure the release with the `configure` script located in this directory. This script tailors various scripts and `make` files to the local software platform. Eventually we hope to provide a more intelligent GNU-style configuration script. For now, you must provide the required information about the platform "by hand". The usage of is `configure` is

   ```
   configure [-mpiinc=MPI_INC_DIR] [-mpilib=MPI_LIB]
             [-f90=F90]
             [-f90com=F90_COM] [-f90lib=F90_LIB]
             [-c++com=C++_COM] [-c++lib=C++_LIB]
             [-lnkcom=LNK_COM]
   ```

   Here

   `MPI_INC_DIR` is the directory where MPI include files are located.

   `MPI_LIB` is the file (or list of files) containing the MPI libraries.

   `F90` is a symbol identifying the Fortran 90 platform. Currently it must be one of `nag` or `aix`.

   `F90_COM` is the command to execute the Fortran 90 compiler

   `F90_LIB` is the file (or list of files) containing Fortran 90 run-time libraries.

`C++_COM` is the command to execute the C++ compiler.

`C++_LIB` is the file (or list of files) containing C++ run-time libraries.

`LNK_COM` is the command to execute the linker.

This release includes various wrapper scripts called `configure.`*host* which illustrate the options we used for to configure the software for various local platforms. You may be able to choose the wrapper script for the host most similar to your own, and edit it. See section 5 for further details. Section 6 provides a more complete guide to the choice of `configure` options.

4. If `configure` completed successfully, build everything by typing

```
make
```

# 4 Using HPF

Add the `bin` directory in the HPF home directory to your path, eg

```
set path = ($path /local/shpf/bin)
```

and the `man` directory to you `man`-path, eg

```
setenv MANPATH ${MANPATH}:/local/shpf/man
```

The command

```
man hpf
```

tells you how to use the `hpf` script to compile an HPF program (this information is also provided in the accompanying release notes).

The output of the compilation is an executable containing calls to the MPI library. The procedure for executing this program depends on your MPI platform.

As an example, suppose HPF has been configured to use a P4-based installation of MPICH. If `test.hpf` is an HPF program requiring 3 processors for its execution, it might be compiled and run by the commands

```
hpf test.hpf
mpirun -mr_pg -machinefile ~/machines -mr_np 3 a.out
```

Because this particular platform was used frequently during development of the HPF system, this release includes a short script called `hpfrun` in the `bin` directory. The call to `mpirun` can be replaced by

```
hpfrun 3 a.out
```

The example assumed you have a file `machines` in your personal home directory, containing a list of host names upon which MPI processes can be spawned. If you want to allow a maximum of 4 MPI processes to be run on just one workstation called `bacchus`—the `machines` file might contain the lines

```
bacchus
bacchus
bacchus
bacchus
```

(We emphasize that `hpfrun` is *not* a portable mechanism for running HPF programs; if your platform is different from the one referred to above you will probably have to provide your own script for this purpose.)

# 5 Platforms on which Southampton HPF has been installed

| Fortran | C++ | MPI | architecture | host |
|---|---|---|---|---|
| Nag f90 | GNU g++ | MPICH/P4 | RS6000 | ben |
| Nag f90 | GNU g++ | MPICH/P4 | SUN 4 | bacchus |
| AIX xlf90 | AIX mpCC | MPICH/EUI | SP2 | sp2 |

Table 1: Some installation platforms

Table 1 lists a set of platforms on which the translation system has been installed. They are tagged by host names. The release directory contains a set of wrapper scripts called `configure.`*host* which illustrate the options we used for these platforms. If your software platform is the same as any of the ones above, it should be possible to construct an appropriate set of `configure` options with minimal changes to the corresponding wrapper—for example, you may only need to change the MPI home directory path name.

# 6 More on configuring

Configuration can be a tedious process, due to the need to get the three major components of the software platform—Fortran 90, C++ and MPI—interworking smoothly. If you experience problems, especially if your platform is significantly different from any of those listed in section 5, please contact us at `info-hpf@par.soton.ac.uk` or `bug-hpf@par.soton.ac.uk`. The information in this section is provided for completeness—it may or may not be useful to typical installers. In later releases we intend to provide a more intelligent configuration script, but first we have to gain experience manually configuring the system on as many platforms as possible.

## What the configure script does

It creates the following files in the HPF home directory

```
src/adlib/Makefile
src/interface/Makefile
bin/hpf
bin/ad++
```

They are constructed from templates, by filling in details about the local software platform.

It also creates an empty directory, `lib`.

## Options to configure

The syntax of the `configure` script was given in section 3. This section goes into more detail on the choice of the various fields in the options for the configure script—their effects, and typical values.

Currently, most of the options are compulsory.

### -mpiinc=MPI_INC_DIR

`MPI_INC_DIR` is the directory where MPI include files are located. This option affects compilation of the HPF run-time system, which occurs when the HPF installation is being built.

*EXAMPLE:* if the MPI installation is MPICH and the MPI home directory is `/local/mpich`, the `MPI_INC_DIR` field should be set to `/local/mpich/include`.

### -mpilib=MPI_LIB

`MPI_LIB` is the file (or list of files) containing the MPI libraries. These libraries are passed to the linker by the `hpf` script. (The `-l`*name* form should also work, provided the linker command knows where to find the named library—see the section on `LNK_COM`, below.)

*EXAMPLE:* if, again, the MPI installation is MPICH, and the MPI home directory is `/local/mpich`, and MPICH is layered on P4 on an RS6000 workstation, the `MPI_LIB` field should probably be set to `/local/mpich/lib/rs6000/ch_p4/libmpi.a`.

### -f90=F90

`F90` is a symbol identifying the Fortran 90 platform. This option affects compilation of the HPF run-time system, which occurs when the HPF installation is being built.

This flag compromises portability, because only a limited number of platforms are recognised. It is needed for the following reason: the HPF run-time library, through which the generated code interfaces to MPI, is written in C (and C++). The C version of the MPI initialisation procedure requires access to the command line arguments passed to the executable at run-time—`argc` and `argv`. But the code generated by the HPF translator is Fortran—the Fortran calls the C. The C has to pick up the argument information somehow. It gets it from the Fortran run-time library. Presently C code appropriate to a particular Fortran platform is compiled in conditionally, according to the value of `F90`.

*EXAMPLES:* if the Fortran 90 platform is NAG f90, `F90` should be set to `nag`. If it is AIX xlf90, `F90` should be set to `aix`. If you have any other Fortran 90 platform, please contact us at `info-hpf@par.soton.ac.uk` or `bug-hpf@par.soton.ac.uk`.

### -f90com=F90_COM

`F90_COM` is the command to execute the Fortran 90 compiler. This command is invoked by the `hpf` script to compile the code generated by the translator.

The command will be applied to a single, automatically-generated Fortran 90 file, with file suffix `.f`. The code in this file is in *fixed* source format. The `F90_COM` command may have to include options to specify this format.

The compiled code will be linked with C run-time library functions, written on the assumption that Fortran subroutine names are presented to the linker after conversion to lower case, with an underscore character appended. The `F90_COM` command may have to include options to specify that it treats procedure names this way.

*EXAMPLES:* if the Fortran platform is NAG f90, a likely value for `F90_COM` is `"f90 -mismatch -w"`. The first option disables some over-zealous checking of procedure interfaces, which can prevent the generated code from compiling; the second option just switches off some associated warning messages. If it is AIX xlf90, a likely value for `F90_COM` is `"xlf90 -qfixed -qextname"`. The first options specifies that the source format is fixed; the second specifies the treatment of procedure names described above.

### -f90lib=F90_LIB

`F90_LIB` is the file (or list of files) containing the Fortran 90 run-time libraries. These libraries are passed to the linker by the `hpf` script. The `-l`*name* form will also work, provided the linker command knows where to find the named library—see the section on `LNK_COM`, below.

The location of these libraries may be documented in the manual page for the compiler; otherwise, they can sometimes be discovered by watching the compiler operate, with a "verbose" option specified.

*EXAMPLE:* if the Fortran platform is NAG f90, a likely value for the `MPI_LIB` is `"/usr/local/lib/f90/libf90.a -lm"`.

### -c++com=C++_COM

`C++_COM` is the command to execute the C++ compiler. This option affects compilation of the HPF run-time system, which occurs when the HPF installation is being built.

The command will be applied to a single C++ file with file suffix `.C`.

*EXAMPLE:* if the C++ platform is GNU gcc, a likely value for the `C++_COM` is `g++`.

### -c++lib=C++_LIB

`C++_LIB` is the file (or list of files) containing the C++ run-time libraries. These libraries are passed to the linker by the **`hpf`** script.

The comments about the Fortran library are applicable, but note that if the C++ compiler is also used as the linker (a likely choice) this option can be omitted, because then the "linker" already knows about the C++ libraries.

### -lnkcom=LNK_COM

`LNK_COM` is the command to execute the linker. This command is invoked by the **`hpf`** script.

The linker must link together compiled Fortran 90 and C++ programs, their run-time libraries, and the MPI library. Because C++ may make special demands on its linker, we have found that the easiest thing is to use the C++ compiler itself as the linker.

*EXAMPLE:* if the C++ platform is GNU gcc, a likely value for the `LNK_COM` is `g++`. On the IBM SP2 we have used the parallel C++ compiler `mpCC`—it was then necessary to specify an import list for linking with AIX Fortran: the value for `LNK_COM` was `"mpCC -bI:/usr/lpp/xlf/lib/lowsys.exp"`