
Compilation and the Environment



- The Environment
- Compilers
- Sequential Compilation
- MPI Compilation
- Shared Memory Compilation
- Other Compilation Considerations

- Access is via ssh only

```
ssh userid@login.hpcx.ac.uk
```

- File Transfer available via sftp

```
sftp userid@login.hpcx.ac.uk
Connecting to login.hpcx.ac.uk...
userid@login.hpcx.ac.uk's password:
sftp> put example.tar.gz
Uploading example.tar.gz to /l1f41jfs/userid/example.tar.gz
sftp> bye
```

- scp also available

```
scp localfile userid@login.hpcx.ac.uk:hpcxdir/hpcxfile
scp hpcxfile  userid@localmachine.ac.uk:localdir/localfile
```

- New users will be prompted to change their password
 - passwords expire after six months
 - passwords must be at least 6 characters long, contain 2 non-alphanumeric characters
- Shells
 - Default is the Korn shell (ksh)
 - Available (supported) shells
 - bsh, csh, ksh, bash
 - To change shells permanently you will need to contact the helpdesk
 - can use a new shell for each login, eg by typing `bash`
- Editors
 - Available editors
 - vi, emacs

- **XL Fortran for AIX**
 - currently version 9.1
 - f77, f90, f95, with extensions
 - 32 and 64 bit support
- **C for AIX**
 - currently version 7.0
 - ANSI C89 compliant compiler
 - 32 and 64 bit support
- **VisualAge C++ for AIX**
 - currently version 7.0
 - fully compliant ANSI 98 C++ compiler
 - 32 and 64 bit support
- **To determine the current compiler version**

```
lslpp -l | grep Compiler
```
- **or**

```
xlf90 -qversion
```

- The IBM Fortran compiler (**xlf**) is invoked via a set of aliases:
 - **xlf** – f77 compilation, fixed form source
 - **xlf90** – f90 compilation, free form source
 - **xlf95** – f95 compilation

- For example:

```
xlf90 -qsuffix=f=f90 -o example example.f90
```

- by default compilers expect source files to have **.f** extension (**.F** for preprocessing)
- **-qfixed=72, -qfree=f90** changes required source form behaviour

- The C compiler has four basic invocations
 - xlc - the main (ANSI) C compiler
 - cc - extended language level for legacy code
 - c89 - strict conformance to ANSI C standard
 - xlc128 - increases long doubles to 128 bits
- The C++ compiler has similar invocations
 - xlc - the basic C++ compiler
 - xlc128 - the C++ equivalent of xlc128
- C++ compiler treats **.c** files as ANSI C by default
- For example

```
xlc -o example1 example1.c
xlc -o example2 example2.C
```

Compilation Options

- Threadsafe compilers
- Datatype Sizes
- Memory issues and 64 bit addressing
- Optimisation
- Other useful options
- Debugging
 - addressed in lecture on tools

- All compilers come in two versions: `normal' and threadsafe.
- Threadsafe (or reentrant) versions have `_r` at the end of the name
 - e.g. `xlf_r` `mpcc_r`
- Threadsafe compilers use stack allocation for local variables.
 - no impact on performance
 - might expose bugs
- In general we recommend using the threadsafe version, especially if installing libraries.

Type (bytes)	IBM	Cray	Altix
complex	2 x 4	2 x 8	2 x 4
double complex	2 x 8	2 x 8	2 x 8
double precision	8	8	8
pointer	4	8	4
integer / logical	4	8	4
real	4	8	4

- Fortran 90 `KIND` syntax - portable alternative
- `-qrealsize=8` - promotes reals to 8 bytes (note that other datatypes are also promoted)
- `-qintsize=8` - promotes integers and logicals to 8 bytes

Type (bytes)	IBM	Cray	Altix
char	1	1	1
short	2	4	2
int	4	8	4
long	4	8	4
float	4	4	4
double	8	8	8
long double	8	8	16

- **xlc128** compiler increases long double size to 16 bytes

- The compilers have limits on the stack and data regions
 - options `-bmaxstack` and `-bmaxdata`
- Both compilers use 32 bit addressing by default
 - maximum datasize 2Gb, stacksize 256Mb
- 64 bit addressing achieved using `-q64` option
 - maximum data and stack size much greater

- For example:

```
xlf90_r -qsuffix=f=f90 -q64 -bmaxstack:0x10000000  
-bmaxdata:0x70000000 -o example example.f90
```

- sets datasize to 1792Mb, stacksize to 256Mb

- **-q64** increases
 - pointers to 8 bytes (Fortran)
 - long datatypes to 8 bytes (C)

- **-O0 (Fortran only)**
 - Fast compilation and full debugging support
- **-O2**
 - low level optimisation
 - e.g. redundant code elimination, invariant code removal from loops, some loop unrolling + pipelining,
 - consider using `-qmaxmem=-1` to allow more memory for space intensive optimisations
 - This is the default for optimisation > 2
- **-O3**
 - more extensive optimisation
 - e.g. some floating point reordering, deeper loop unrolling, ...
 - consider using `-qstrict` to avoid numerical differences

- **-O4** includes all the optimisations of **-O3**, plus:
 - **-qhot** (for fortran), **-qipa**, **-qarch=auto**,
-qtune=auto, **-qcache=auto**
- **-qhot**
 - Higher Order Transformations
 - optimises F90 array syntax statements
 - changes loop nests to improve cache behaviour
 - allows user to
 - transform loops to vector library calls
 - introduce array padding
 - Syntax
 - **-qhot[=[no]vector | arraypad[=n]]**

- **-qipa**
 - can expand the scope of optimisation to an entire program unit
 - `-qipa[=level=n | inline=name | fine tuning]`
- **-O5**
 - includes all of -O4 optimisations plus: `-qipa=level=2`
- **Target machine options**
 - specifies a particular processor and memory system
 - `-qarch`: restricts compiler to a particular instruction set
 - `-qtune`: optimisation based on a particular machine
 - `-qcache`: defines a particular cache / memory geometry
 - options: `pwr4`, `pwr5`, `auto`, ...

- **-qinitauto**
 - initialises variables to zero at compile time
- **-u**
 - equivalent to `IMPLICIT NONE` in your code
- **-qlanglvl**
 - use to check code against nonconformance with Fortran, C or C++ standard
- **I/O Buffering**
 - can be controlled using the `XLFRTEOPTS` environment variable
 - `export XLFRTEOPTS buffering=disable_all`
 - or you can use the `flush_` routine
 - for C you can use `setvbuf`

- MPI codes may be compiled using:
 - `mpxlf90_r`
 - `mpcc_r`
 - `mpCC_r`
- These scripts invoke the `xlf_r`, `xlc_r` and `xlC_r` compilers and link in the MPI libraries
- For example

```
mpxlf90_r -qsuffix=f=f90 -o example example.f90
```

```
mpcc_r -o example example.c
```

- 64 bit addressing
 - only the MPI threaded library supports 64-bit applications
- MPI -2
 - includes most of the MPI -2 standard
 - eg single sided communication, MPI -I O
 - except: "Process Creation and Management" functionality
 - e.g. `MPI_COMM_SPAWN`
- **-qrealsize=8** promotes reals to 8 bytes
 - `MPI_REAL` is NOT, however, promoted!

- Shared memory programs (e.g. OpenMP) *must* be compiled using the threadsafe versions:
 - `xlf90_r`
 - `xlc_r`
- OpenMP is supported for Fortran, C (C++ in 7.0)
- Compile with the `-qsmp` option

```
xlf90_r -qsuffix=f=f90 -qsmp=omp -o example
example.f90
```
- **XLSMPOPTS** environment variable allows local thread stack size to be increased

```
export XLSMPOPTS="stack=bytes"
```

Fortran	C
<code>xlf</code>	<code>xlc</code>
<code>xlf_r</code>	<code>xlc_r</code>
<code>xlf90</code>	<code>xlc128</code>
<code>xlf90_r</code>	<code>xlc128_r</code>
<code>xlf95</code>	<code>cc</code>
<code>xlf95_r</code>	<code>cc_r</code>
<code>mpxlf</code>	<code>cc128, cc128_r</code>
<code>mpxlf_r</code>	<code>c89</code>
<code>mpxlf90</code>	<code>mpcc</code>
<code>mpxlf90_r</code>	<code>mpcc_r</code>
<code>mpxlf95</code>	<code>xlc, xlc_r</code>
<code>mpxlf95_r</code>	<code>xlc128, xlc128_r</code>

- A series of example Makefiles are available at
<http://www.hpcx.ac.uk/support/FAQ/template.tar>

- Mixed MPI / OpenMP programming
- SHMEM codes
- Java
- HPF

- Mixed MPI / OpenMP codes are possible on the IBM system
- As with OpenMP, you must use the thread-safe library
- For example:

```
mpxlf90_r -qsuffix=f=f90 -qsmp=omp -o example  
example.f90
```

- TurboSHMEM exists for the Power 5 architecture
 - allows SHMEM codes to be ported quickly
 - actual performance will depend on particular routines used
 - uses both MPI and LAPI (Low-Level Application Programming Interface)
 - almost complete SHMEM implementation
- Libraries
 - libsmac, libsmaf, libsmac64, libsmaf64
- For example

```
xlf90_r -qsuffix=f=f90 -o example example.f90 -lsmaf
```

- **Java**

- IBM J2RE 1.3.1 installed on HPCx
- for example:
 - javac -O example.java
 - java example
- shared memory Java code (Java Native threads, JOMP) run within an node
- MPJ not installed
- IBM J2RE 1.4.2 is also available at `/usr/java14/bin/java`

- **HPF**

- not available on the HPCx system

- Introduced the basic HPCx environment
- Introduced the C, C++ and Fortran compilers
- Considered compilation options for optimisation
- Considered parallel program compilation
- Summarised other relevant compilation options

- The HPCx Service User Guide

<http://www.hpcx.ac.uk/support/documentation>

- Compiler information

www.software.ibm.com/ad/fortran

www.software.ibm.com/ad/caix

www.software.ibm.com/ad/vacpp

- Redbooks

<http://www.redbooks.ibm.com/>