

# Computer Programming Languages and Unix Power Tools

By Mark Mc Keown

ITC Research Computing Support Group

[res-consult@virginia.edu](mailto:res-consult@virginia.edu)

# Topics

- Compiled Languages
- Compilers
- Integrated Development Environments
- Debuggers
- Scripting Languages
- Unix Tools

# General Programming Advice

- ALWAYS document your code.
- Use DEBUG statements which can be removed by the Pre-processor.
- When developing your code compile across a number of architectures to insure the code is portable and bug free.
- Be prepared to re-write the code if necessary.
- Do not re-invent the wheel.
- Keep functions/sub-routines short (approx one screen of text)
- White space and blank lines are Free.

# More General Programming Advice

- Compile with full warnings on – (e.g. if using gcc use the `-Wall` option).
- Today's warnings are tomorrows bugs/errors.
- Use descriptive names for variables (e.g. `NoPeople` rather than `N`)
- Read more then one book on the language you are using.
- Read other peoples code.
- Use `lint` or `ftnchek` to check your code.
- If learning a language from a book do some of the exercises.

# Debug Statements

## Fortran

Parameter DEBUG=1

```
if ( DEBUG.eq.1) then  
  WRITE(2,*) #####  
endif
```

## C/C++

```
#define DEBUG
```

or

```
cc -DDEBUG -c test.c
```

```
#ifdef DEBUG  
  printf(“#####”);  
#endif
```

# General Programming References

- Programming Pearls by Bentley.
- The Pragmatic Programmer : From Journeyman to Master by Hunt et al.
- The Practice of Programming by Kernighan and Pike.
- Code Complete: A Practical Handbook of Software Construction by McConnell.

[http://www.itc.virginia.edu/research/petsc\\_docs/code\\_management.html](http://www.itc.virginia.edu/research/petsc_docs/code_management.html)

# Compiled Languages

- Fortran
- C
- C++
- Java
- C#

# Fortran (Formula-Translation)

- Popular versions of Fortran used are: Fortran 77, Fortran 90 and Fortran 95 (Fortran 2000 will be released soon)
- Easy to learn.
- Very efficient.
- Supports multi-dimensional arrays (Matrices).
- Lots of legacy codes are written in Fortran.
- Lots of numerical libraries such as IMSL and NAG exist.
- Not used much outside academia
- Supports Complex Numbers

# Fortran References

- Fortran 90/95 for Scientists and Engineers by Stephen Chapman
- Fortran 90/95 Explained by Metcalf and Reid
- Fortran 77 for Engineers and Scientists with an Introduction to Fortran 90 by Nyhoff and Leestma
- Programming with Fortran 77 by Mayo & Cwiakala
- Numerical Recipes in Fortran by Press et al.
- comp.lang.fortran newsgroup

# Fortran Advice

- Use Fortran 90 – it has many advantages over Fortran 77
- Avoid if possible goto
- Use “`IMPLICIT NONE`”

# Fortran Webpages

- High Performance Fortran

<http://www.crpc.rice.edu/HPFF/>

- <http://www.netlib.org/liblist.html>

- <http://www.lahey.com/other.htm>

- Ftnchek – a Fortran 77 program checker

<http://www.dsm.fordham.edu/~ftnchek>

- <http://www.polyhedron.com>

- <http://www.itc.virginia.edu/research/fortranprog.html>

- <http://www.itc.virginia.edu/research/u015.fortran.html>

<http://www.itc.virginia.edu/research/compiler>

- IBM `xlf`            `-O3 -qstrict -lmass -qhot -qarch  
-qtune`
- Sun `f77/f90`        `-fast -xcrossfile`
- SGI `f77/f90`        `-Ofast`
- GNU `g77`            `-O3 -ffast-math -funroll-loop`

# Fortran Compilers for Windows

Compaq Visual Fortran: (\$519 with IMSL, \$389 without)

<http://www.compaq.com/fortran/>

Absoft: (\$499 with IMSL, \$299 without)

<http://www.absoft.com>

Intel Fortran Compiler – must be used with MS Visual Studio (\$499)

<http://developer.intel.com/software/products/compilers/f50/>

# Fortran Compilers for Linux

- g77 – free, but does not support Fortran 90 and may not be as efficient as a commercial compiler.

- Portland Group Compilers (\$239 - \$559)

<http://www.pgroup.com/>

- Absoft (\$375 with IMSL)

<http://www.absoft.com>

# C

- Popular versions are: K&R C and ANSI C, new standard was formalized in 1999.
- Relatively easy to learn.
- Efficient - but not as fast as Fortran.
- Similar syntax to Java and C++.
- Used widely outside of Academia.

# C References

- [The C Programming Language](#) by Kernighan and Ritchie
- [C Programming: A Modern Approach](#) by KN King
- [comp.lang.c](#)
- [comp.lang.c.moderated](#)
- <http://www.eskimo.com/~scs/cclass/cclass.html>
- Lint – a C program checker  
<http://www.pdc.kth.se/training/Tutor/Basics/lint/index-frame.html>

# C Compilers

- IBM x1C      `-O3 -qstrict -qtune=pw2sc`
- SGI CC      `-Ofast`
- Sun CC      `-fast -xcrossfile`
- gcc      `-O2 -ffast-math`

# Microsoft Visual C++

- \$45 for Academic version in the UVa bookshop
  - \$21 for Visual C++ Pro through Microsoft
- Select at UVa:

<http://www.itc.virginia.edu/licenses/selectmain.htm>

- Do not use MFC if you intend to port to Unix.
- It is possible to use the Intel C++ compiler with Visual C++ - the Intel compiler should produce faster binaries (cost \$399)

<http://developer.intel.com/software/products/compilers/c50/>

# C++

- ISO standard was ratified in 1998 – however different compilers support the standard to different levels which can lead to portability problems
- C++ is a large language which can take time to learn.
- Can be as efficient as Fortran but it is difficult to acquire this level of performance.
- Supports object-orientated programming.
- Supports Meta-programming with Templates.
- Used widely outside Academia.
- Supports complex numbers

# C++ References

- Beginner:

[Accelerated C++](#) by Andrew Koenig

- Intermediate:

[The C++ Programming Language](#) by Stroustrup

- Advanced:

[Effective C++](#) by Meyers

[More Effective C++](#) by Meyers

[Modern C++ Design](#) by Alexandrescu

# C++ Web References

- `comp.lang.c++`
- `comp.lang.c++.moderated`
- `comp.lang.c++.std`
- <http://www.research.att.com/~bs/C++.html>
- <http://www.zib.de/Visual/people/mueller/Course/Tutorial/tutorial.html>
- <http://www.cs.wustl.edu/~schmidt/C++/>
- <http://www.mysteries-megasite.com/linux/C-tutorial.html>

# C++ Toolkits

- Many Toolkits have been written in C++ such as Blitz++ and Pooma.
- A good reference site for Scientific programming tools in C++ is:

<http://www.oonumerics.org/>

# ITC Courses on C++

- ITC have a 12 hour course on C++ run over four evenings – the course is free:

<http://www.itc.virginia.edu/training/student/programming.htm>

# KAI C++ Compiler

- The KAI compiler is available on most Unix systems – helps portability.
- Produces very fast and efficient binaries.
- Converts C++ code to C and compiles with the native compiler.
- Available on a lot of National Supercomputers.
- \$320 dollars for an academic Linux License.
- [www.kai.com](http://www.kai.com)

# Java

- Programming Language developed and owned by Sun Microsystems - proprietary software.
- Easy to learn.
- Write once run anywhere. (or write once debug everywhere !!)
- Very slow compared to Fortran.
- Object Orientated.
- Garbage Collection.
- Very Popular in Industry.
- Microsoft's J++ != Java

# Java Reference

- [Java How to Program](#) by Deitel and Deitel
- [Core Java 2, Vol 1&2](#) by Horstmann and Cornell
- `comp.lang.java.programmer`
- `comp.lang.java.*`
- <http://java.sun.com/>

# C#

- Simpler than C or C++
- Similar Syntax to C/C++
- Object Oriented.
- Garbage Collection.
- Part of Microsoft's .NET framework

# Debuggers

- dbx – can be used interactively, in batch mode or analysis core files.
- dbx is found on most Unix systems

<http://www.itc.virginia.edu/research/u029.dbx.html>

- gdb - is the GNU version of dbx for gcc, g++ and g77

# Integrated Development Environments

- An IDE contains an editor, a debugger and performance tuning tools in a single product with a simple to use GUI.

- **SGI Workshop:**

invoked with **cvmain**

<http://www.sgi.com/developers/devtools/index.html>

- **Sun Workshop:**

invoked with **workshop**

<http://www.sun.com/forte/developer/>

# Scripting Languages

- Generally easier to write than compiled languages.
- Very powerful – lots of in built functionality.
- Rapid Development.

Perl

Python

# Perl

- Originally designed to address the shortcomings of Unix Scripts.
- Very powerful for handling text.
- Popular for CGI programming.
- Supports Object Orientated programming.
- “More than one way to skin a cat”
- Can be difficult to read.

# Perl References

- [Learning Perl](#) by Schwartz
- [Programming Perl](#) by Larry Wall et al.
- <http://www.perl.org/>
- <http://www.perl.com/>
- `comp.lang.perl.misc`

# Perl Courses by ITC

- Perl Programming Introduction (\$175)

<http://www.itc.virginia.edu/training/courses/spec-perl-prog-intro.html>

- Perl Programming for the Web (\$200)

<http://www.itc.virginia.edu/training/courses/spec-perl-prog-web.html>

# Python

- Very easy to learn
- Often used to prototype ideas that are to be coded in a compiled language.
- Supports numerical types such as int, float – also supports arbitrary precision.
- Supports Object Orientated programming.
- Supports complex numbers.

# Python References

- [Learning Python](#) by Lutz et al.
- [Programming Python](#) by Lutz et al.
- <http://www.python.org/>
- `comp.lang.python`

# Python Add On Tools

- NumPy - Numerical extensions to Python, includes an interface to LAPACK and FFTPACK.

<http://www.python.org/topics/scicomp/numpy.html>

- PyMat – a Matrix package for Python – uses Matlab

<http://claymore.engineer.gvsu.edu/~steriana/Python/pymat.html>

- SWIG – generates Python wrappers for C++ libraries

<http://www.swig.org/>

# ITC Python Course

ITC are running an 11 week course on Python beginning Sept 10. The classes run from 1pm to 5pm on Monday afternoons. The cost is \$200. For more information e-mail

[Stormy@virginia.edu](mailto:Stormy@virginia.edu)

# SILOON

- Scripting Interface Languages for Object-Oriented Numerics
- Allows programmers to easily access existing object-oriented scientific frameworks and numerical libraries written in C, C++, and Fortran.
- Programmers use scripting languages to glue together components with interpreted run-time scripts.

<http://www.nersc.gov/ACTS/siloon/main.html>

# Unix Editors

- Vi

<http://www.itc.virginia.edu/desktop/unix/docs/u004.vi.html>

Learning the vi Editor by Lamb et al.

- emacs

<http://www.itc.virginia.edu/research/emacs.html>

[http://www.itc.virginia.edu/research/petsc\\_docs/codemanagement.html](http://www.itc.virginia.edu/research/petsc_docs/codemanagement.html)

Learning Gnu Emacs by Cameron et al.

# More Unix Editors - GUI

- Xemacs

[www.xemacs.org](http://www.xemacs.org)

- nedit

[www.nedit.org](http://www.nedit.org)

<http://www.itc.virginia.edu/research/nedit.html>

- Pico

<http://www.indiana.edu/~ucspubs/b103/>

- Jove

<http://www.itc.virginia.edu/desktop/unix/docs/u003.jove.html>

# Unix Tools

- Shells
- Useful Commands
- Pipes & Redirects

# Shells

- sh, csh, ksh, tcsh, bash, zsh
- Recommend tcsh or bash for interactive use. Both have command completion, simple command line editing and simple to use history facilities.
- Change logon shell using chsh
- Recommend sh shell for shell scripts

# Useful Commands

- flip
- grep
- sed
- awk
- make
- top
- find
- sort
- which
- nohup
- Wildcards

# Pipes & redirects

- Pipes are used to pass the output from one Unix command as the input to another Unix command.

```
ls | grep "mmk"
```

- Redirects are used to pass the output of a Unix command into a file.

```
ls > directory_listing
```

# Unix Shell Scripts

- It is possible to save a set of Unix commands in a file to execute in a batch mode – such a file is called a Shell Script.

# Unix References

- Learning the Unix Operating System by Peek et al
- Unix in a Nutshell by Robbins et al.
- The Unix Programming Environment by Kernighan et al.
- Unix Power Tools by Peek et al.
- ITC Web Pages on Unix:  
[www.itc.virginia.edu/desktop/unix/docs/home.html](http://www.itc.virginia.edu/desktop/unix/docs/home.html)

# ITC Unix Courses

- Unix Course (12 hours - Free)

<http://www.itc.virginia.edu/training/student/programming.htm>

- Linux/Unix Workshop - 44 class hours over 11 days, cost \$225

<http://www.itc.virginia.edu/training/courses/spec-linux-unix.intro.html>