

Programming languages

Outline

- Why bother with programming?
- Interpreted v compiled languages
- Object oriented programming
- Scripting
- Compiling commands and makefiles
- Useful libraries
- Solaris and Linux availability

Why write a program?

- Existing software doesn't do the job...
 - ... well enough (new algorithm)
 - ... fast enough
- You're doing something completely new so no-one thought to write the code yet!
- You want to put it on your CV !

- So which is the best language to use ...?

Interpreted v compiled

- **Interpreted** languages

- Are converted to machine language 'on the fly'
- Must be converted each time code is executed
- Allow rapid code development
- Tend to run slowly, so not good for number crunching
- Error checking can't be so complete

- **Compiled** languages

- Are completely converted to machine language before execution
- Conversion happens only once
- More tedious edit-compile-run-edit cycle
- Make full use of the CPU available
- Better memory management
- Can have very full error checking

Language types

- **Procedural** languages
 - Traditional languages where code is a sequence of instructions executed (approximately) in sequence
 - BASIC, FORTRAN
 - Code can become 'spaghetti' for large tasks unless procedural methods are added
 - PASCAL, C
- **Object-oriented** languages
 - Objects contain data and the methods to work on that data
 - Extensive use of message passing
 - C++, Java, FORTRAN90
 - Good for large projects due to concepts such as encapsulation and inheritance

Which languages matter?

- FORTRAN
- FORTRAN90
- ~~BASIC~~
- C
- ~~ADA~~
- C++
- ~~C#~~
- JAVA
- IDL
- ~~Prolog~~
- ~~COBOL~~
- ~~PASCAL~~
- ~~Delphi~~
- Perl
- TCL
- Python
- ~~Assembly~~
- ~~Smalltalk~~
- ~~LISP~~

FORTRAN / F90 / F95

- Designed for scientific programming
- Efficient for numerical work
- Huge amount of code exists
- Good for conversion to parallel environments

- COMMON blocks, computed GOTOs are crashes waiting to happen
- No dynamic memory handling
- No low-level hardware access (playing with bytes tricky)
- Often code relies on 'standard' extensions to the language (which are not standard at all!)

- FORTRAN90 adds object-oriented programming to a language never intended to have it!

C

- Flexible – a general-purpose language
 - Efficient for everything
 - Huge amount of code exists
 - Complex data structures are simple to create
 - Very portable
 - Huge range of libraries/add-ons available
-
- Lowish-level - a single command doesn't do much
 - Can be hard to read
 - C lets you do bad things, C++ is pickier
 - 'pointers' are a source of confusion for beginners

C++

- C with added object-oriented capabilities
- Usual benefits of OOP
 - Good for large collaborative programming projects
- Standard Template Library provides complex objects
 - vectors
 - maps
 - stacks
 - algorithms
- Easy to write 'C with objects'
- Can be hard to read
- C lets you do bad things, C++ is pickier
- 'pointers' are a source of confusion for beginners

Java

- Completely object-oriented
- C++ with the nasty/interesting bits taken out
 - No operator overloading
 - No multiple inheritance
 - No templates
- Very portable due to 'virtual machine'
- Can create graphical applications (or web applets) relatively easily

- Slow due to virtual machine
- Not much scientific programming done in it
- Keeps changing!

IDL

- High-level language for data analysis
- Powerful built-in statistical tools
- Excellent plotting & image display
- Commercial & expensive

- Semi-interpreted, so not so fast
 - looping very slow
- Danger of using 'black-box' routines
 - e.g. the Gaussian fitting is very weak

Example 1

- Print “Hello World” 10 times
- Components
 - Initialisation
 - Loop creation
 - Printing
 - Tidying up

Example 2

- Calculate sum of 10x10 array of integers
- Components
 - Creation/initialisation of 2-D arrays
 - Looping in two dimensions
 - Filling with random numbers
 - Printing results

Scripting

- Example of why scripting helps:
 - I have a program which reads a file containing 100 integers, sums them, and prints the answer
 - I have 200 files like that, and I want to process them all
- Do I ...
 - (a) run my program 100 times, typing in a different file name each time?
 - (b) change the program so it reads a list of files and processes them each in turn
 - (c) write a script which feeds each file in turn to the existing program

Scripting languages

- C-shell (csh)
 - the standard scripting language for UNIX
 - uses the same commands you'll use on the command line
 - only integer arithmetic
 - Text handling is tricky

- PERL
 - A special case... a scripting language which is (almost) as powerful as a programming language
 - C-like syntax
 - You can add modules to extend the language
 - PerlTk – a GUI toolkit
 - PGPLOT – for plotting
 - FITSIO – handling astronomical FITS files
 - Excellent for reading output from programs, extracting results from the output, and re-formatting data.
 - Used in cgi scripting on WWW pages
 - poor passing of values into functions and subroutines
 - if you need subroutines, it's probably time to think again about a script!

Compiling, linking and executing

- Compiling (convert source code to object code)
 - C ... `gcc -c mycode.c`
 - C++ ... `g++ -c mycode.cpp`
 - FORTRAN ... `g77 -c mycode.f`
- Linking (combining object code and libraries)
 - `gcc -o myprog -lmylibrary mycode.o` (links to `libmylibrary.a`)
- Compile and link in one stage
 - `gcc -o myprog -lmylibrary mycode.c`
- Executing the code
 - `./myprog` ... `./` means 'from the current directory'
- Makefiles are a set of instructions which can compile and link complex projects
 - save typing multiple long complex commands
 - only rebuild things that have changed

Useful libraries

- Numerical recipes (books and code library)
 - Equation solving
 - Fitting/minimisation
 - Random number generation (built-in functions are all poor!)
- SLALib (positional astronomy library)
 - co-ordinate transforms
 - tangent plane projections
 - MJD/epoch conversions
 - some fitting routines
 - <http://star-www.rl.ac.uk/star/docs/sun67.htx/sun67.html>
- NAG (numerical algorithms group)
 - as numerical recipes, but commercial
 - a good solution, IF you can understand the manual
- cfitsio/DAL
 - direct access to FITS files
- astrolib
 - big library of IDL procedures for all things astronomical