

Function Pointers ○○○	Using Existing Code ○○○○○○	Parallel Computing ○○○○○○○○○○○○○○○○	Other Programming Languages ○○○	Summary ○
--------------------------	-------------------------------	--	------------------------------------	--------------


C for Science

Lecture 5 of 5

Sam Bott

<http://wwwf.imperial.ac.uk/~shb104/c>
s.bott@imperial.ac.uk

15th February 2014



Sam Bott
C for Science

Function Pointers ●○○	Using Existing Code ○○○○○○	Parallel Computing ○○○○○○○○○○○○○○○○	Other Programming Languages ○○○	Summary ○
--------------------------	-------------------------------	--	------------------------------------	--------------

Last Week...

- Arrays in C are not limited to two dimensions.
- We can create our own data types using **structs**.
- Global variables can be accessed anywhere in your program.
- The lifetime of a variable can be extended by making it **static**.
- A 'Release' build will optimise your program during compile-time.
- Using the bits that make up individual data types can be an efficient alternative the data types themselves.

Sam Bott
C for Science

Function Pointers ○●○	Using Existing Code ○○○○○○	Parallel Computing ○○○○○○○○○○○○○○○○	Other Programming Languages ○○○	Summary ○
--------------------------	-------------------------------	--	------------------------------------	--------------

Pointers IV - Functions Pointers

More Pointers!

- So far, we have seen examples of pointers to:
 - native types (such as `int*`, `float*`, `double*`...)
 - something unknown, which we *can* recast (`void*`)
 - structures (such as `FILE*`)
 - and pointers themselves (such as `int**`, `double**`...)
- There is one more pointer type, a pointer to a *function*.

Why point to functions?

C code is often re-used between projects. Mathematical routines such as ones to find roots of functions, become extremely limited if they are tied down to a specific case.

Declaring a pointer to a function

We do this with a `typedef`:

```
typedef double (* fx) (double x);
```

Sam Bott
C for Science

Function Pointers ○○●	Using Existing Code ○○○○○○	Parallel Computing ○○○○○○○○○○○○○○○○	Other Programming Languages ○○○	Summary ○
--------------------------	-------------------------------	--	------------------------------------	--------------

Example of a Function Pointer

Function Pointers - An example

```
#include <stdio.h>
typedef double (* fx) (double x);

double newtonSolve(fx func, fx grad, double guess)
{
    unsigned int loop;
    for (loop = 0; loop < 10; loop++) guess -= func(guess)/grad(guess);
    return guess;
}

double sample(double x)
{
    return x*x-2.0;
}

double dsample(double x)
{
    return 2.0*x;
}

int main()
{
    double sol = newtonSolve(sample, dsample, 1.0);
    printf("Solution = %.15lg\n", sol);
    printf("Residue = %lg\n", sample(sol));
    return 0;
}
```

This should get you started with exercise # 5.

Sam Bott
C for Science

Don't Do it All Yourself!

- As you have seen from the exercises, writing functions to solve equations such as cubics can become complicated (especially when rounding error needs to be minimised).
- A lot of people have spent a considerable amount of time attempting to perfect implementations of mathematical functions.
- Routines written by a third party are packaged in *libraries*.
- These are available for you to link your program with.

Using Fortran Code

- In addition to libraries, there is existing source code that can be used...
- Scientific programming has been going on for ≈ 70 years (counting from Colossus) in Britain.
- Unfortunately a lot of this has been carried out in Fortran.

f2c

Bell Labs have released a program called `f2c` which converts Fortran source code to C. The output from `f2c` is usually not very human friendly, but it does at least compile.

GNU Scientific Library - GSL

GNU have released their C Scientific Library:

<http://www.gnu.org/software/gsl/>

- It is managed by scientists at Los Alamos, and is very comprehensively documented.
- GSL requires gcc (but “ports” are available - Windows users, see the handouts on my web page)

Licensing

“GSL can be used internally (‘in-house’) without restriction, but only redistributed in other software that is under the GNU GPL.”

GSL Example

```
#include <stdio.h>
#include <gsl/gsl_sf_bessel.h>

int main()
{
    double x = 0.0;

    printf("Enter x \n");
    scanf("%lf", &x);
    printf("J0(%g) = %.18e\n", x, gsl_sf_bessel_J0(x));

    return 0;
}
```

Compile this using the command-line (assuming you have GSL installed):

```
gcc gsl-sample.c -lgsl -lgslcblas -lm -o gsl-sample
```

Function Pointers
○○○
Using Existing Code
○○○○●○
Parallel Computing
○○○○○○○○○○○○○○○○○○○○
Other Programming Languages
○○○
Summary
○

Scientific Libraries

NAGLIB

The Numerical Algorithms Group, based in Oxford, maintain a software library called NAGLIB.

- Certain departments in Imperial College have a license for this.
- Check with the Software Shop; you might be covered.

Sam Bott
C for Science

Function Pointers
○○○
Using Existing Code
○○○○●○
Parallel Computing
○○○○○○○○○○○○○○○○○○○○
Other Programming Languages
○○○
Summary
○

Scientific Libraries

General Numerical Software



The Netlib Repository contains a lot of very useful numerical code and papers. It is definitely worth having a route through their website:

www.netlib.org

Sam Bott
C for Science

Function Pointers
○○○
Using Existing Code
○○○○○○○
Parallel Computing
●○○○○○○○○○○○○○○○○○○○○
Other Programming Languages
○○○
Summary
○

Parallel Computing

Parallel Computing

What is Parallel Processing?

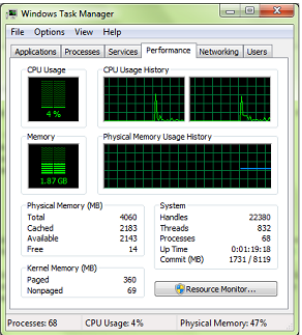
- The programs we have been creating compile to machine code which is passed (serially) through a processor.
- Modern desktop computers no longer have just one processor with one processing core within it. They have one or more, multi-cored processors.
- This means there is the ability to run multiple 'threads' of a program, each one on its own core, allowing us to take advantage of the full power of the processor.
- There are a few ways to utilise multiple processors, but it can't be done for all code/algorithms.

Sam Bott
C for Science

Function Pointers
○○○
Using Existing Code
○○○○○○○
Parallel Computing
●○○○○○○○○○○○○○○○○○○○○
Other Programming Languages
○○○
Summary
○

Parallel Computing

How many processors do I have?



In Windows press
[Ctrl] + [Shift] + [Esc]
to open Task Manager, then look at the "Performance" tab.

In Linux run:

```
grep -c "^processor" /proc/cpuinfo
```

Sam Bott
C for Science

Ways to utilise Multiple Processors

Wherever our code is running a loop (i.e. doing the same thing across a large data range) we can start to think about parallel execution. I'm going to mention three of the many ways to take full advantage of your processor:

- 1 Running multiple instances of the program.
- 2 Running threads in a shared memory environment.
- 3 Using MPI in a distributed memory environment.

1. Running Multiple Instances

- Running multiple instances of a serial program is the simplest way to use more than one processor.
- We simply write our code as normal and start n instances of it at the same time, with n being the number of cores to use.
- If your program is running across multiple discrete datasets - for example, processing multiple output files from another program - then running multiple instances, each one targeting separate datasets, might be effective.
- This is only appropriate when no data needs to be shared between processes.

2. Threads in Shared Memory

- Within a program, one can spawn multiple execution threads.
- Threads are all owned by the parent process but are running independently. They can each access their own and shared memory.
- Threads can be spawned arbitrarily, but are particularly useful for splitting up parts of `for` loops between cores and running them in parallel.
- Because running loops in parallel means that they no longer execute sequentially, only algorithms where an iteration does not rely on another can be run in this way.
- Threads in shared memory (not surprisingly) require shared memory. As a "rule of thumb" this is available anywhere where the processors sit in the same physical box.
- We will look at simple use of OpenMP to make our programs run in this way.

3. MPI in Distributed Memory

- Multi-threaded code for shared memory could alternatively be coded for distributed memory.
- A Distributed Memory System is one in which not all processors are addressing the same memory. A common scenario would be using more than one node in a computation cluster.
- Parallel code for distributed memory requires using a Message Passing Interface to send information between the threads/processes.
- This is often more difficult as (unlike with OpenMP) the initial single-threaded code needs to be restructured.
- Programs using MPI can be used on shared and distributed memory systems alike.
- Using MPI is beyond the scope of this course, but you may wish to investigate it for future projects.

Function Pointers ○○○	Using Existing Code ○○○○○	Parallel Computing ○○○○○○●○○○○○	Other Programming Languages ○○○	Summary ○
--------------------------	------------------------------	------------------------------------	------------------------------------	--------------

Multi-Threading with OpenMP

OpenMP

OpenMP (Open Multi-Processing) is an API (Application Programming Interface) that can be used to create a multi-threaded program in shared memory with only minor modifications to your initial, single-threaded program.

It is included with the majority of compilers. In `gcc` use the `-fopenmp` switch to enable OpenMP. For Visual Studio or Code::Blocks see the information sheets on my web page.

Multi-threading, and the use of OpenMP, are large topics. We will just cover the use of OpenMP to run `for` loops in parallel; this should be sufficient to create efficient, parallel code for a large number of applicable algorithms.

Sam Bott
C for Science

Function Pointers ○○○	Using Existing Code ○○○○○	Parallel Computing ○○○○○○●○○○○○	Other Programming Languages ○○○	Summary ○
--------------------------	------------------------------	------------------------------------	------------------------------------	--------------

Multi-Threading with OpenMP

Writing Multi-Threaded Code

- 1 Write your algorithm, running on a single thread.
- 2 Compile, debug and thoroughly test you code.
- 3 Identify regions that could run in parallel - look for `for`s!
- 4 Add OpenMP compiler directives.
- 5 Compile with OpenMP support.

Sam Bott
C for Science

Function Pointers ○○○	Using Existing Code ○○○○○	Parallel Computing ○○○○○○●○○○○○	Other Programming Languages ○○○	Summary ○
--------------------------	------------------------------	------------------------------------	------------------------------------	--------------

Multi-Threading with OpenMP

When Can We Multi-Thread?

When one iteration does not depend on another.

Bad Examples

```

8 for(i = 1; i <= n; ++i)    //Loop 1
9   a[i] = a[i-1] + b[i];

5 for(i = 0; i < n; ++i)    //Loop 2
6   x[i] = x[i+1] + b[i];

```

The two above examples could not be run in parallel:

- The first example requires the previous iteration to be finished before it can run the next.
- The second could not be run in parallel as another thread might change the original value of $x[i+1]$ before it has been used to calculate $x[i]$.

Sam Bott
C for Science

Function Pointers ○○○	Using Existing Code ○○○○○	Parallel Computing ○○○○○○○○●○○○	Other Programming Languages ○○○	Summary ○
--------------------------	------------------------------	------------------------------------	------------------------------------	--------------

Multi-Threading with OpenMP

Using OpenMP

To make a `for` loop run on multiple cores, use the `parallel for` OpenMP compiler directive on the line above:

```

6 #pragma omp parallel for
7 for (i = 0; i < max_i; i++)
8 {
9     a[i] = b[i] + c[i+1];
10 }

```

This will automatically split up the `for` loop, and run it on all available cores.

Sam Bott
C for Science

Function Pointers ○○○	Using Existing Code ○○○○○	Parallel Computing ○○○○○○○○○○○○○○○	Other Programming Languages ●○○	Summary ○
--------------------------	------------------------------	---------------------------------------	------------------------------------	--------------

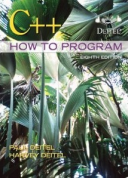
C++

Moving on to C++

What is it?

- C++ can be thought of very loosely as an object oriented extension to C.
- The C++ standard is over twice as big as the C standard.
- C++ is still actively developed and maintained.

References



- One good book that has been brought to my attention is “C++ How to Program, 8th Edition” by Harvey and Paul Deitel published in 2011.
- Before buying any C++ book, make sure that it is recent, and that you like the writing style.

Sam Bott
C for Science

Function Pointers ○○○	Using Existing Code ○○○○○	Parallel Computing ○○○○○○○○○○○○○○○	Other Programming Languages ○●○	Summary ○
--------------------------	------------------------------	---------------------------------------	------------------------------------	--------------

C#

Moving on to C#

C# belongs to a different class of language.

- C# targets the .NET framework. This can be thought of as an **extensive** library of well-maintained and (mostly) stable general-purpose functions.
- .NET runs in software, rather than directly on hardware. This results in slower code, but a consistent and safe execution environment.
- C# is *my* language of choice. The language, together with .NET, favours fast and intuitive development of powerful programs.
- Pointers are hidden from the user in C#. Most memory allocation and de-allocation is done automatically, via *garbage collection*.
- To combine the execution speed of C and the development power of C# I have previously written fast algorithms into C libraries and written a program around it in C#.
- C# is less portable than C; C# can only be run on .NET for windows, or Mono for Linux, Mac, Android and iOS.

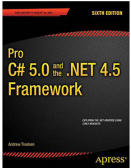
Sam Bott
C for Science

Function Pointers ○○○	Using Existing Code ○○○○○	Parallel Computing ○○○○○○○○○○○○○○○	Other Programming Languages ○●○	Summary ○
--------------------------	------------------------------	---------------------------------------	------------------------------------	--------------

C#

C# References

Book



- “Pro C# and the .NET 4.5 Framework, 6th Edition” by Andrew Troelsen, appears to be up-to-date and comprehensive.

MSDN

The definitive (and free) source of information for Microsoft Platforms is the MSDN, the C# documentation is browsable at:

<http://msdn.microsoft.com/en-gb/library/kx37x362.aspx>

Sam Bott
C for Science

Function Pointers ○○○	Using Existing Code ○○○○○	Parallel Computing ○○○○○○○○○○○○○○○	Other Programming Languages ○○○	Summary ●
--------------------------	------------------------------	---------------------------------------	------------------------------------	--------------

Summary

- We can create pointers to functions. This enables us to pass functions to another function.
- There exist function libraries that you can use rather than write your own versions of established algorithms.
- By running multiple instances of your program, or writing a multi-threaded program, you can use the full processing power of your computer.
- An HPC is a very powerful computer-processing cluster and can be hired to run CPU intensive code.

Sam Bott
C for Science