COMP1917: Computing 1

1. Introduction

Reading: Moffat, Chapter 1.

©UNSW, 2012

Course Web Site

`http://www.cse.unsw.edu.au/~cs1917/14s2`

Please check this Web Site regularly for updated information, including:

- Course Outline
- Notices
- Tutorial and Lab Exercises
- Assignments
- Style Guide
- Sample Programs
- Supplementary Material

©UNSW, 2012

Textbook

Recommended Text: Alistair Moffat, *Programming, Problem Solving, and Abstraction with C*, Pearson Educational, Australia, 2003.

Reference Texts: Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, 2nd Edition, Prentice Hall, 1988.

Jeri R. Hanly and Elliot B. Koffman, *Problem Solving and Program Design in C, 7th Edition*, Addison-Wesley, 2013.

Paul Davies, *The Indispensable Guide to C: With Engineering Applications*, Addison-Wesley, 1995.

©UNSW, 2012

Occupational Health and Safety (OHS)

- Computer Ergonomics and OHS for Students

  `www.cse.unsw.edu.au/about-us/help-resources/for-students/ergonomics`

- Other Resources for Students

  `www.cse.unsw.edu.au/about-us/help-resources/for-students`

©UNSW, 2012

Plagiarism

- ALL work submitted for assessment (including labs and assignments, etc.) must be your own work
- Collaborative work in the form of "think tanking" is encouraged, but students are not allowed to derive code together as a group during such discussions
- Plagiarism detection software is used on submitted work
- See Yellow Form:
  `www.cse.unsw.edu.au/people/studentoffice/policies/yellowform.html`
  Unix Primer:
  `http://newcse.cse.unsw.edu.au/help/doc/primer/primer.pdf`
  CSE Addendum to UNSW Plagiarism policy:
  `www.cse.unsw.edu.au/~chak/plagiarism/plagiarism-guide.html`

©UNSW, 2012

Planned Topics

1. Introduction
2. Numbers In, Numbers Out
3. Making Choices
4. Loops
5. Functions
6. Binary and Hexadecimal
7. Number Storage and Accuracy
8. Characters and Arrays
9. Pointers
10. Strings and Files
11. Writing a Makefile
12. Debugging
13. Structures
14. Linked Lists
15. Stacks and Queues
16. Binary Search Trees
17. Memory and Stack Frames
18. Machine Language
19. Sorting and Efficiency

©UNSW, 2012

Old Lectures on YouTube

- Richard Buckland's COMP1917 Lectures from Session 1, 2008 are available on `YouTube.com`
- they are are not a requirement, but may be used as a supplementary resource, to see some of the topics presented from a different perspective
- syllabus was somewhat different, and topics were covered in a different order
- Search for "COMP1917 UNSW"

©UNSW, 2012

Guiding Philosophy

In this course we would like to:

- encourage the use of abstraction to solve problems.
- emphasise good documentation and coding practice.
- gain a general understanding of how a computer system works.

©UNSW, 2012

Beyond COMP1917

This course provides you with the foundation necessary to deal with more complex concepts in:

- COMP1927: Computing 2
- COMP2911: Engineering Design in Computing
- COMP2041: Software Construction
- COMP2121: Microprocessors and Interfacing

©UNSW, 2012

Why C ?

- good example of an imperative language
- widely used in industry (and science)
- many libraries and resources
- fast compilers
- provides low level access to machine

©UNSW, 2012

Brief History of C

- C and UNIX share a complex history.
- C was originally designed for and implemented on the UNIX operating system on a PDP-11 computer.
- Dennis Ritchie was the author of C (around 1971).
- In 1973, UNIX was rewritten in C.
- BCPL strongly influenced the C language.
- B (author: Ken Thompson, 1970) was the predecessor to C, but there was no A.

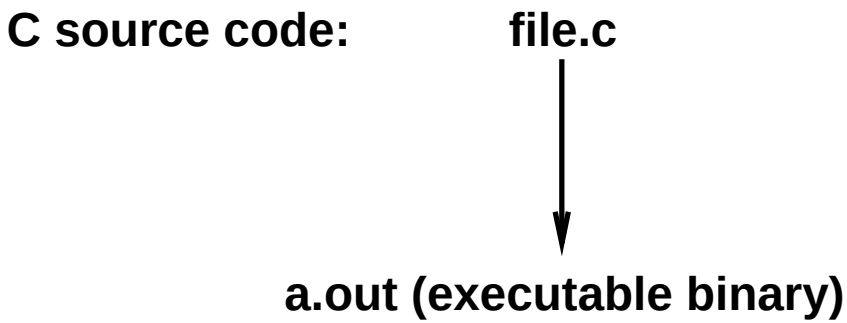
©UNSW, 2012

Brief History of C *cont.*

- BCPL and B were both typeless languages.
- C is a *typed* language (but not strongly).
- In 1983, American National Standards Institute (ANSI) established a committee to clean up and standardise the language.
- In 1988, the ANSI C standard was published.
- This greatly improved source code portability.
- C is the main language for writing operating systems and compilers, but it is also commonly used for a variety of applications.

©UNSW, 2012

Getting Started: C Compilation—one file

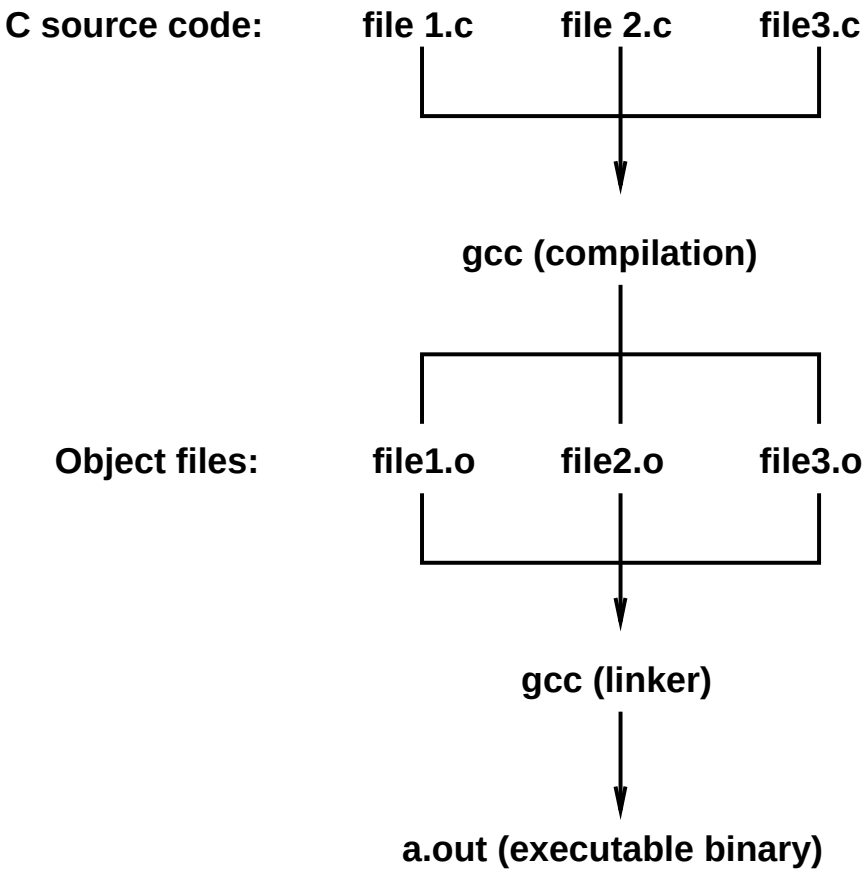


To compile `file.c`, you type the following:

```
gcc file.c
```


©UNSW, 2012

Getting Started: C Compilation—many files



©UNSW, 2012

Your First Program

The traditional *Hello World* program.

```
main( void )
{
    printf( "Hello, World!\n ");
}
```

*Note: This example is **not** fully ANSI compliant.*

©UNSW, 2012

Compiling with `gcc`

To compile `hello.c`, you type the following:

```
gcc hello.c
```

To run the program type:

```
./a.out
```


©UNSW, 2012

Command line Options with `gcc`

- The default with `gcc` is to compile your program uncritically, i.e. without giving you any warnings about potential problems.
- Good practice is to be tough on yourself:

```
gcc -Wall hello.c
```

- The "-Wall" option tells `gcc` to report all warnings to anything it finds that is potentially wrong and non ANSI compliant.
- Having the compiled program in `a.out` is not always helpful:

```
gcc -o hello hello.c
```

- The `-o` option tells `gcc` to place the compiled object file in the named file rather than `a.out`

©UNSW, 2012

Compiling with `dcc`

Tip: if you are having trouble tracking down some nasty bugs in your code, try compiling it with `dcc` instead of `gcc`. `dcc` does the same job as `gcc`, but it does more checking for memory overlaps, etc. and will (hopefully) help you to find and fix the nasty bugs.

©UNSW, 2012

Breakdown of *Hello World*

```
// include standard library defs and functions
#include <stdio.h>

int main( void ) // function and entry point
{
    // library call to print string constant
    printf( "Hello, World!\n" );

    return 0;   // tell OS that no error occurred
}
```


©UNSW, 2012

Key Points

- Functions (subroutines) can be called anything except “ `main` ”, which is special.
- `main` is where the program starts executing.
- There must be one (and only one) `main`
- Functions can have parameters (`printf`) or no parameters (`main`).
- Curly brackets “{ }” are used to enclose statements in a function.
- A function can be called by naming it in a statement.
- Each statement is terminated by a semicolon “;”

©UNSW, 2012

Key Points *cont.*

- Comments are denoted by "`//`" or with a pair of "`/*`" and "`*/`".
- "`//`" comments do not extend beyond the end of line.
- Comments between "`/*`" and "`*/`" can span multiple lines.

Example:

```
/* This is a library function that
   is defined in the header file stdio.h */

printf("Hello World\n");  // this calls the function
```


©UNSW, 2012

Basic Structure of a C Program

```
// include files
// global definitions
// global variables

// function definitions
int f( int x )
{
    // local variables

    // body of function

    return( ... );
}

.
.
.
```

```
.
.
.
// main function
int main( )
{
    // local variables

    // body of main function

    return 0;
}
```


©UNSW, 2012