Programming in C

Objectives

- Learn the core elements of ANSI C
- Get to know tools for building C programs on GNU/Linux
- Learn to read and debug C programs
- Build useful utilities for reusing in your own programs

Introduction

History

- Originally designed and implemented by Dennis Ritchie on DEC PDP-11 [1]
- Influenced by B [2] written by Ken Thompson in 1970
- First C standard in 1988 by ANSI (C89)
  - Adopted by ISO in 1990 (C90)
- Most recent standard C99 by ISO [3]
- Compiled language
- Source code portable

C program – hello.c

```
#include <stdio.h>

 /* function main - print hello world */
int
main()
{
  printf("hello world!\n");
  return 0;
}
```

C program structure

- Multi-line comments begin with `/*` and end with `*/`, these are called delimiters
- `#` is used to begin pre-processor directives
- Execution of a C program begins at function main
  - main can return an `int` value to the operating system otherwise it should return `void`
- Code blocks and function bodies begin with `{` and end with `}`
- C statements end with `;`

Executing hello.c

- Use GCC [4]

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

  – Without the o option the executable is a.out

- Execute

```
./hello
```

- Output

```
hello world!
```

Compilation process

- A compiler produces the executable by performing through the following steps
  - Pre-processing
  - Compilation and assembly
  - Linking

Pre-processing

- Conceptual first step in compilation
- Two tasks commonly performed
  - File inclusion
    - `#include` directive

```
#include <stdio.h>
```

  - Macro substitution
    - `#define` directive

```
#define pf printf
pf("hello world!")
```

Compilation and assembly

- Lexical and semantic analysis to generate intermediate code
- Transform the intermediate code to assembly or machine code
- Creating an object file using GCC

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

  - The `c` option tells GCC to not perform linking
  - A file called `hello.o` is produced

Linking

- Linking combines all the object files and required library code to produce a single executable

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

Multiple source files – hello.c

```
#include "print.h"

 /* Function main - Print hello world */
int
main()
{

  print_hello("hello world!\n");

}
```

Multiple source files – print.c

```
#include <stdio.h>
#include "print.h"

void print_hello()
{
  printf("printing: hello
  world!");
}
```

Multiple source files – print.h

```
#ifndef _PRINT_H_
#define _PRINT_H_

extern void print_hello();

#endif //_PRINT_H_
```

- `#ifndef` - `#endif` sequence above serves to guarantee that the pre-processor does not include the same header file twice in a source file.

Simple compilation

- Executing gcc

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

- Executing hello

```
./hello
```

- Output

```
printing: hello world!
```

Complex Compilation- An Error

- hello.c
  `gcc hello.c -c`
  – produces `hello.o`
- print.c
  `gcc print.c -c`
  – produces `print.o`
- Error?

```
gcc hello.o -o hello
hello.o(.text+0x27):hello.c: undefined
  reference to `_print_hello'
collect2: ld returned 1 exit status
```

Complex Compilation – Correcting the Error

- Using `gcc`

  ```
  gcc hello.o print.o -o hello
  ```

- Using `ld` [5] (`gcc` with -v switch shows how)

  ```
  ld -o hello /lib/crt0.o -
    L/opt/gcc.3.3/lib/gcc-lib/i586-pc-
    interix3/3.3 hello.o print.o -lgcc -lc -
    lpsxdll -v
  ```

  - `/opt/gcc.3.3/lib/gcc-lib/i586-pc-interix3/3.3` is the path to `libgcc.a` in my Windows SFU installation

Debug Using DDD

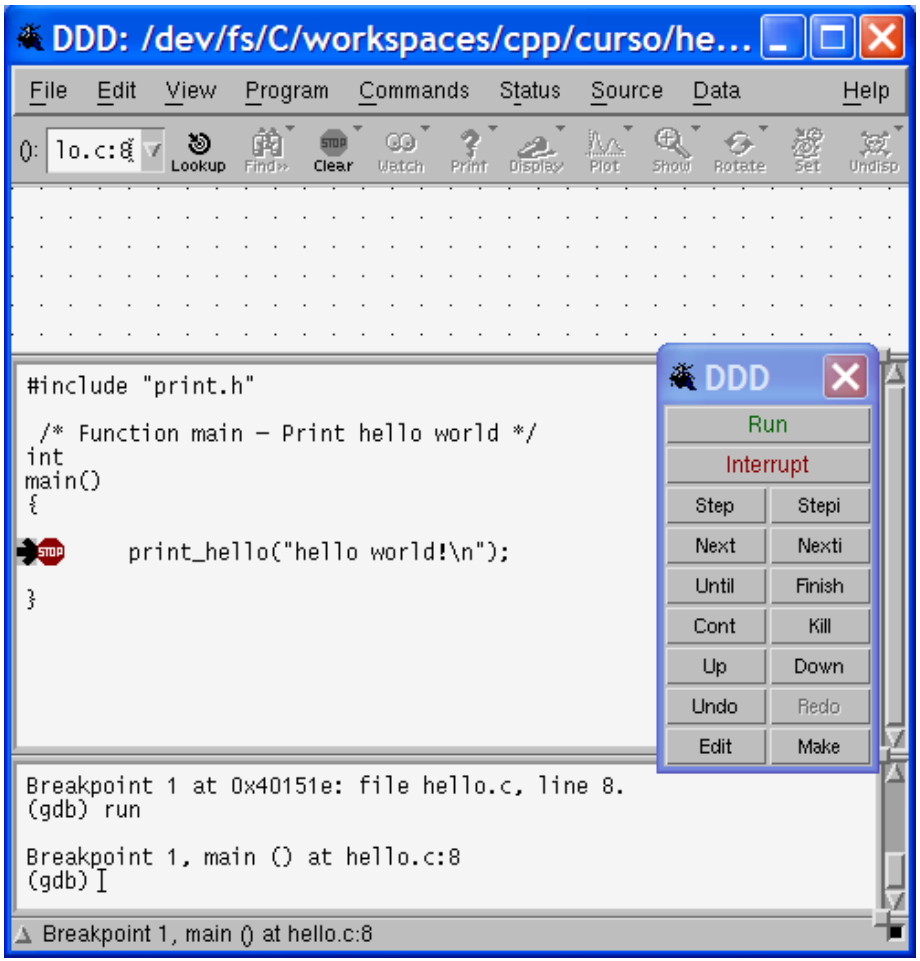
- DDD is a graphical debugger for X Windows and it uses gdb, the command line debugger
- Re-compile source code with extra debug information for gdb

```
gcc hello.c print.c -o
  hello –g
```

- Execute ddd

```
ddd hello
```

- Try stepping through code and adding watch expressions



Other Topics

- Creating static and shared libraries [6]
- Dynamic linking
- GCC compile, link and optimize options
- Building applications with make
- Using Eclipse and CDT [7] for C/C++ development

References

1. The Development of the C Language - http://cm.bell-labs.com/cm/cs/who/dmr/chist.html
2. THE PROGRAMMING LANGUAGE B - http://cm.bell-labs.com/cm/cs/who/dmr/bintro.html
3. JTC1/SC22/WG14 - C - http://www.open-std.org/jtc1/sc22/wg14/
4. GCC - http://gcc.gnu.org/
5. GNU Binutils - http://www.gnu.org/software/binutils/
6. Shared vs static libraries - http://www.linuxselfhelp.com/HOWTO/GCC-HOWTO/x575.html
7. Eclipse CDT - http://www.eclipse.org/cdt/