C – Practical Tools

Paul Schrimpf

June 1, 2007

References

These notes go over the simplest way to compile a C program. For more information see:

- ACM Developing Software in Unix Tutorial – everything you need to get started in Unix or Linux
- GCC manual

Getting a Compiler

- Many compilers are available – `gcc` is the most common, Intel's `icc` can be faster, many others exist
- To install
  - Linux / Unix – already have it
  - Windows
    - ★ IDE – Bloodshed Dev-C++
    - ★ Unix-like – MINGW, Cygwin
  - Mac OS X – Xcode
- If you just want to use within Matlab, try typing `mex`, it may work
- If you don't use an IDE, you'll want a quality editor too – Emacs and Vi(m) are classics
- My preference: Linux or MINGW & Emacs

Hello, World!

```
1 #include <stdio.h>
2
3 int main () {
4   printf("Hello, World!\n");
5 }
```

1. Open your editor, type it, and save it as helloWorld.c
2. To compile: `gcc helloWorld.c -o hello.exe`
3. To run: `./hello.exe`

- Fun fact: A Tutorial Introduction to the Language B (Kernighan, 1972) – contains the original "Hello World!"

Hello, Matlab!

```
1 #include "mex.h"
2
3 void mexFunction(int nlhs, mxArray *plhs[],
4                  int nrhs, const mxArray *prhs[])
5 {
6   mexPrintf("Hello, Matlab!\n");
7 }
```

1. To compile: `mex helloWorld.c` (within Matlab)
2. To run: `helloWorld()` (within Matlab)

Hello, Stata!

```
1 #include "stplugin.h"
2
3 STDLL stata_call(int argc, char *argv[])
4 {
5     SF_display("Hello Stata!\n");
6     return(0);
7 }
```

1. To compile: `gcc -ansi -shared -fPIC [-DSYSTEM=OPUNIX] stplugin.c helloStata.c -o hello.plugin`
2. To use, in stata:
   ```
   program hello, plugin
   plugin call hello
   ```

Hello What?

```
1  int i;main(){for(;i["i"]<i;++i){--i;}"];read('-'-'-',i+++"hell\
2  o, world!\n","/"/"/"));}read(j,i,p){write(j/p+p,i---j,i/i);}
```

- Winner of the 1984 International Obfuscated C Code Contest

Compiler Options

- Compiling can get very complicated
  - Hundreds of options – `man gcc` is 6884 lines
  - Linking libraries
  - Breaking into steps
- Tools to manage
  - IDE – automates much, menus for options
  - `make` – manages complicated compilations A Make Tutorial
- Important options:
  - `-g` turns on debugging information
  - `-Wall` turns on all warning messages
  - `-l`*lib* tells to link against library lib – e.g. for math, `-lm`
  - `-O2` and `-O3` adds optimizations – can do even better with fine tuning

Debugging

- Debuggers
  - Within IDE
  - gdb – command line, or better yet, within emacs (`<Alt-x> gdb`)
    - ★ GDB Tutorial
  - More debuggers
- `lint` – like Matlab's `mlint`

Memory Related Errors

- Memory bugs – leaks, illegal addressing – are very difficult to diagnose
- Effect of memory bugs can depend on program inputs, compiler options, length of program execution, etc.
- Responsible for many computer viruses
- See Techniques for memory debugging
- Tools:
  - Valgrind – indispensible Intro to Valgrind
  - Electric Fence