High-Performance Computing: Compiling and Executing Programs Under Linux

Adrian F. Clark: alien@essex.ac.uk

2016–17

Interpreted code

For *interpreted* languages (Python, Perl, Ruby, Tcl, *etc*), the easiest way to run it is by explicitly invoking the interpreter

```
python myprog.py
```

Alternately, if you make the first line of the program tell the Unix shell where to find the interpreter

```
#!/usr/bin/env python
```

and you have made the program executable by typing

```
chmod +x myprog.py
```

(you have to do this only once) then you can run it just like any other command

```
./myprog.py
```

Note that the `.py` extension is *not necessary* under Linux

Compiling and running C

For *compiled* languages (C, C++, Java, *etc*), you have to compile the program to executable form before running it

```
gcc -g -o myprog myprog.c -lm
./myprog
```

For Java, you need to run it under the interpreter explicitly

```
java myprog.class
```

or write a one-line "wrapper" shell script

Remember that programs compiled on a PC (which has an Intel processor) will *not* run on a Raspberry Pi (which has an ARM processor) or *vice versa*

Timing code under Linux

You can time your program using the `time` command

```
time ./myprog
```

You are normally interested in what it reports as "user" time

Compiled languages should be faster than interpreted ones by a factor of at least 10

Using the optimizer

You can usually squeeze a little more speed out of your program by compiling it with the optimizer

```
gcc -O -o myprog myprog.c -lm
```

Optimizers used to be prone to introducing run-time errors, so a wily programmer will ensure that a program compiled `-O` gives the same results as when it compiled without optimization

Editing code

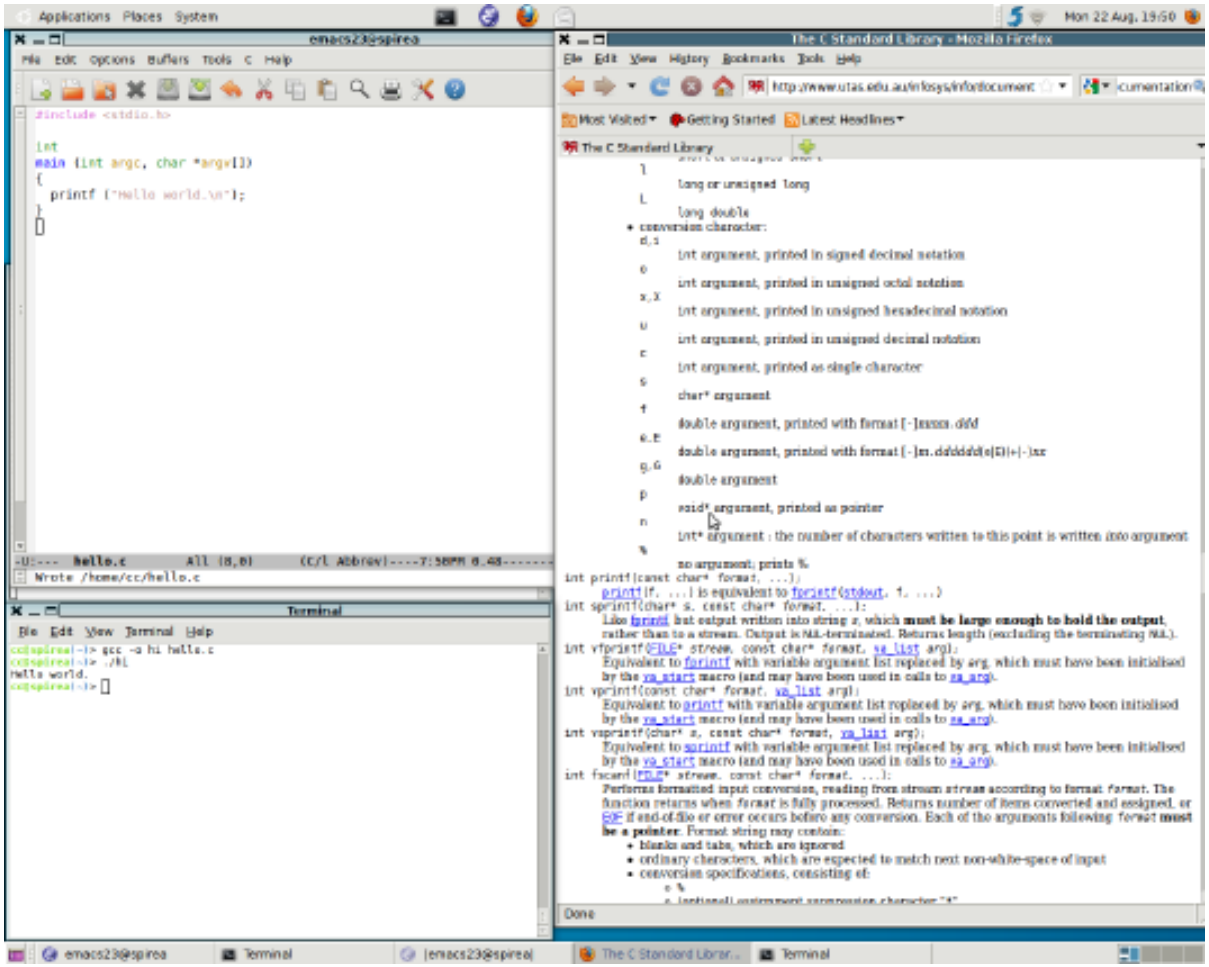

Figure 1:Typical Linux workspace layout

Finding out more

An introduction to program development under Linux is available via ORB:

`http://orb.essex.ac.uk/ce/ce316/dev.html`

This covers provides more detail and covers, editing, using a symbolic debugger, and so on