

Access to GNU/Linux

- all second floor labs are dual boot with Windows 8 and Mint 14 (GNU/Linux)
- to login to Mint you will need to create an account
- this is done by visiting [this web page](http://mcgreg.comp.glam.ac.uk/login.html) `<http://mcgreg.comp.glam.ac.uk/login.html>`
 - fill in the details and submit the query
 - note you can only access that web page from on Campus

Access to GNU/Linux

- then you need to read your email and reply to the automated mail message from mcgreg
- finally you have to wait (about 2 minutes) for another message from mcgreg before you can attempt to login to the Mint clients in the laboratories
- note that you must do all the above within one day or mcgreg will forget about you!
- once done your account will remain until the end of the academic year

How do I change my password?

- if you have forgotten your password then you need to visit [this web page](http://mcgreg.comp.glam.ac.uk/login.html) `<http://mcgreg.comp.glam.ac.uk/login.html>` and follow the advice given on the first slide

- you can use a GUI by clicking on

- **Applications -> System settings -> Users -> yourname**
 - and click on **change password**

How do I change my password?

- if for any reason the above fails you can use the command line.
 - then open a gnome-terminal and type:

```
$ yppasswd
```

- you will be prompted for your old password and asked to type in your new password twice
- once you have done this you can destroy the command line terminal by using the mouse appropriately

Introduction to GNU/Linux

- the most important tools for this course are:
 - g++
 - gdb
 - valgrind
 - gprof

- the editor/ide choice is up to you
 - gedit
 - emacs or vim
 - codeblocks
 - monodevelop

Hello world

- first begin by typing the hello world program into a simple editor (gedit)

```
#include <stdio>

main ()
{
    printf("hello world\n");
}
```



- the GNU C++ compiler, all GCC related compilers have some common options:

- `g++ -g -c -Wall hello.cpp`
 - `-g` means turn on debugging information,
 - `-Wall` turn on all warnings
 - `-c` do not link the object, just compile the source into an object file

- `g++ -pg -c -Wall hello.cpp`
 - add runtime instrumentation to the object

- `g++ -O -Wall hello.cpp`
 - perform optimizing



- `g++ -O0 -Wall -g -c hello.cpp` do not optimize during compilation, and generate an object with debugging info
 - this should be your default during development

- `g++ -O3 -g -Wall hello.cpp`
 - perform maximum optimization, with debugging info
 - do not attempt to debug this, unless you really must, as the code will be transformed considerably



- `g++ -O0 -g -Wall hello.cpp` produce an executable with debugging and no optimization applied
 - should see an `a.out` appear in your current working directory

Running your executable

- during development you should run your program by firstly:

```
$ gdb ./a.out  
(gdb) run  
(gdb) quit
```

- and secondly:

```
$ valgrind ./a.out
```

Running your executable

- only once it runs perfectly, ie no errors were shown from the previous two invocations, should you run it via:

- ```
$./a.out
```