GCC and Make

by: Haotao (Eric) Lai

compile / Link a simple program

```
// hello.c
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

To compile it and generate an executable file using the following command will give you an executable file name “a”.

```
// compile
gcc hello.c

// run it
./a
```

If you don’t want it to be named “a”, you can use `-o` as parameter:

```
// it will give you an executable file name "hello"
gcc -o hello hello.c
```

A few commonly-used GCC compiler options are:

```
$ g++ -Wall -g -o Hello.exe Hello.cpp
```

- `-o` : specifies the output executable filename.
- `-Wall` : prints "all" warning messages.
- `-g` : generates additional symbolic debugging information for use with gdb debugger.

Compile and Link Separately

The above command compile the source file into object file and link with other object files (system library) into executable in one step. You may separate compile and link in two steps as follows:

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

- `-c` : compile into object file “hello.o”. By default the object file name is the same as the source file with extension of “.o”.
- Linking is performed by a program call “LD”, it takes the object file and link them to the other files into an executable file.

Compile and Link Multiple Source Files

However, we usually compile each of the source files separately into object file, and link them together in the later stage. **In this case, changes in one file does not require re-compilation of the other files.**

```
g++ -c file1.cpp
g++ -c file2.cpp
g++ -o myprog.exe file1.o file2.o
```

Compile into a Shared Library

To compile and link C/C++ program into a shared library (".dll" in Windows, ".so" in

Unixes), use -shared option.

At times, it is necessary to use native codes (C/C++) to overcome the memory management and performance constraints in Java. Java supports native codes via the Java Native Interface (JNI). Read “Java Native Interface” for example.