**Compiling with GCC**

**(*main.c*)**

#include <stdio.h>

#include “reciprocal.hpp”

int main (int argc, char \*\*argv)

{

int i;

i = atoi (argv[1]);

printf (“The reciprocal of %d is %g\n”, i, reciprocal (i));

return 0;

}

**(*reciprocal.cpp*)**

#include <cassert>

#include “reciprocal.hpp”

double reciprocal (int i) {

// I should be non-zero.

assert (i != 0);

return 1.0/i;

}

**(*reciprocal.hpp*)**

#ifdef \_\_cplusplus

extern “C” {

#endif

extern double reciprocal (int i);

#ifdef \_\_cplusplus

}

#endif

**Compiling a Single Source File**

To compile a C source file

gcc -c main.c

The C++ compiler is called g++.

g++ -c reciprocal.cpp

g++ -c -I ../include reciprocal.cpp

**Linking Object Files**

g++ -o reciprocal main.o reciprocal.o

g++ -o reciprocal main.o reciprocal.o –lpam

g++ -o reciprocal main.o reciprocal.o -L/usr/local/lib/pam –lpam

**Using *getopt\_long***

const struct option long\_options[] = {

{ “help”, 0, NULL, ‘h’ },

{ “output”, 1, NULL, ‘o’ },

{ “verbose”, 0, NULL, ‘v’ },

{ NULL, 0, NULL, 0 }

};

**Create Archives**

ar cr libtest.a test1.o test2.o

**(*test.c*)**

int f ()

{

return 3;

}

**(*app.c*)**

int main ()

{

return f ();

}

gcc -o app app.o -L. –ltest

**Shared Libraries**

gcc -c -fPIC test1.c

PIC stands for position-independent code.

Then you combine the object files into a shared library, like this:

gcc -shared -fPIC -o libtest.so test1.o test2.o

gcc -o app app.o -L. –ltest

gcc -static -o app app.o -L. –ltest

**Using *LD\_LIBRARY\_PATH***

gcc -o app app.o -L. -ltest -Wl,-rpath,/usr/local/lib

**Library Dependencies**

**(*tifftest.c*)**

#include <stdio.h>

#include <tiffio.h>

int main (int argc, char\*\* argv)

{

TIFF\* tiff;

tiff = TIFFOpen (argv[1], “r”);

TIFFClose (tiff);

return 0;

}

gcc -o tifftest tifftest.c –ltiff

**Dynamic Loading and Unloading**

You could open a shared library named libtest.so by calling dlopen like this

**dlopen (“libtest.so”, RTLD\_LAZY)**

void\* handle = dlopen (“libtest.so”, RTLD\_LAZY);

void (\*test)() = dlsym (handle, “my\_function”);

(\*test)();

dlclose (handle);

Para compilar necesitamos poner -ldl

g++ holaadios.cpp -o holaadios –ldl

**Processes**

**Looking at Processes**

Los procesos hijo heredan las tuberías padre.

Los paréntesis abren un subproceso: (Se duplica la terminal). Por ejemplo: **(cd pepe).**

**Ejecutar el proceso en segundo plano se utiliza al final el &**

**(echo “Hola” && sleep 3 && echo “adios”)&**

Un proceso que se ejecuta en segundo plano se llama **job**, que tienen un number

**Jobs** nos muestra un listado de todos los detenidos y ejecutándose.

Detener un proceso con ctrl + z.

Como levantar un proceso que está en segundo plano (fg y el número del proceso).

**Process IDs**

#include <stdio.h>

#include <unistd.h>

int main ()

{

printf (“The process ID is %d\n”, (int) getpid ());

printf (“The parent process ID is %d\n”, (int) getppid ());

return 0;

}

**Viewing Active Processes**