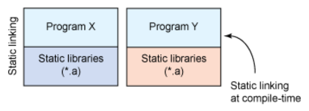
CS 35L Software Construction Lab Week 8 – Dynamic Linking

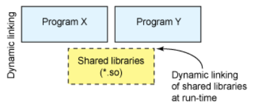
Anatomy of Linux shared libraries

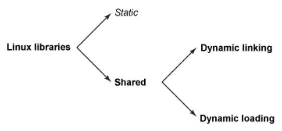
- Libraries - to package similar functionality → modular programming
- Linux supports two types

static library
functionality to bind to a program statically at compile-time

dynamic library
functionality to bind to a program dynamically at run-time







dynamic linking - have Linux load the library upon execution
dynamic loading - selectively call functions with the library in a process

http://www.ibm.com/developerworks/library/l-dynamic-libraries/

Dynamic Loading

to let an application load and link libraries itself

- application can specify a particular library to load, then
- application can call functions within that library

load shared libraries from disk (file) into memory and re-adjust its location
done by a library named ld-linux.so.2

the Dynamic Loading API

dlopen - makes an object file accessible to a program
void *dlopen(const char *file, int mode);
RTLD NOW → relocate now; RTLD LAZY → to relocate when needed;
dlsym - gives resolved address to a symbol within this object
void *dlsym(void *restrict handle, const char *restrict name);
check char *dlerror(); if an error occurs
dlerror - returns a string error of the last error that occurred
dlclose - closes an object file

Creating static and shared libs in GCC

- mymath.h

```
#ifndef _ MY_MATH_H

#define _ MY_MATH_H

void mul5(int
*i);
void add1(int
*i);
#endif
```

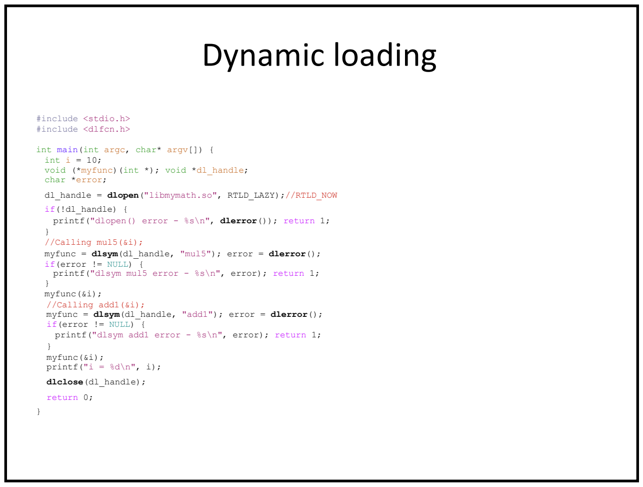
- mul5.c

```
#include
"mymath.h"
void mul5(int
*i)
{

  *i *= 5;

}
```

- add1.c

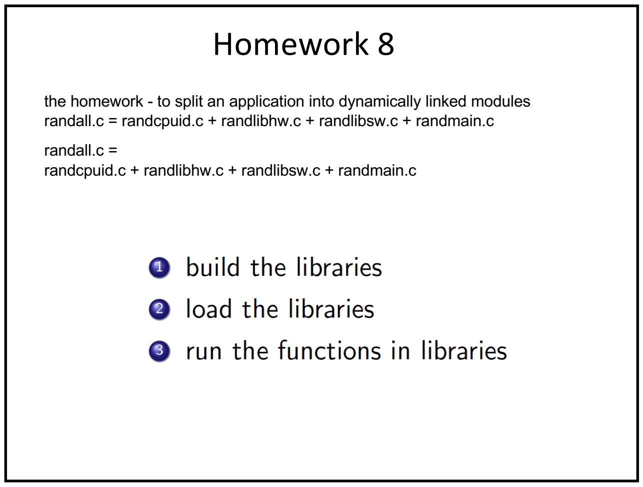
```
#include
"mymath.h"
void add1(int
*i)
{

  *i += 1;

}
```

- gcc -c mul5.c -o mul5.o
- gcc -c add1.c -o add1.o
- ar -cvq libmymath.**a** mul5.o add1.o ----> (static lib)
- gcc -**shared** -fpic -o libmymath.**so** mul5.o add1.o -----> (shared lib)



Dynamic loading

```
#include <stdio.h>
#include <dlfcn.h>

int main(int argc, char* argv[]) {
  int i = 10;
  void (*myfunc)(int *); void *dl_handle;
  char *error;

  dl_handle = dlopen("libmymath.so", RTLD_LAZY);//RTLD_NOW

  if(!dl_handle) {
    printf("dlopen() error - %s\n", dlerror()); return 1;
  }
  //Calling mul5(&i);
  myfunc = dlsym(dl_handle, "mul5"); error = dlerror();
  if(error != NULL) {
    printf("dlsym mul5 error - %s\n", error); return 1;
  }
  myfunc(&i);
   //Calling add1(&i);
   myfunc = dlsym(dl_handle, "add1"); error = dlerror();
   if(error != NULL) {
    printf("dlsym add1 error - %s\n", error); return 1;
   }
   myfunc(&i);
   printf("i = %d\n", i);

   dlclose(dl_handle);

   return 0;
}
```

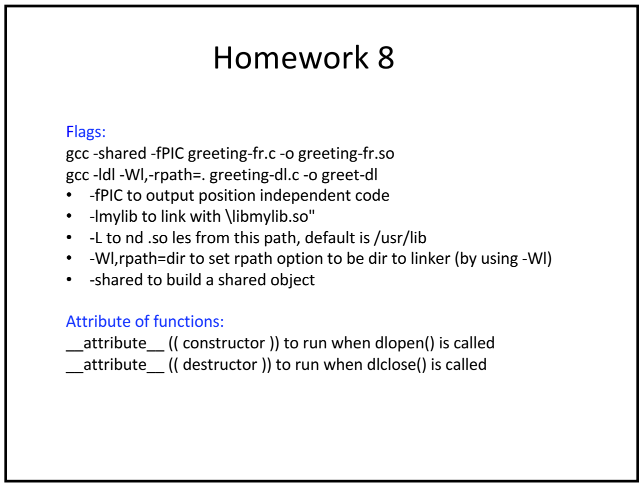


Homework 8

the homework - to split an application into dynamically linked modules
randall.c = randcpuid.c + randlibhw.c + randlibsw.c + randmain.c

randall.c =
randcpuid.c + randlibhw.c + randlibsw.c + randmain.c

1. build the libraries
2. load the libraries
3. run the functions in libraries



Homework 8

Flags:

gcc -shared -fPIC greeting-fr.c -o greeting-fr.so
gcc -ldl -Wl,-rpath=. greeting-dl.c -o greet-dl

- -fPIC to output position independent code
- -lmylib to link with \libmylib.so"
- -L to nd .so les from this path, default is /usr/lib
- -Wl,rpath=dir to set rpath option to be dir to linker (by using -Wl)
- -shared to build a shared object

Attribute of functions:

__attribute__ ((constructor)) to run when dlopen() is called
__attribute__ ((destructor)) to run when dlclose() is called