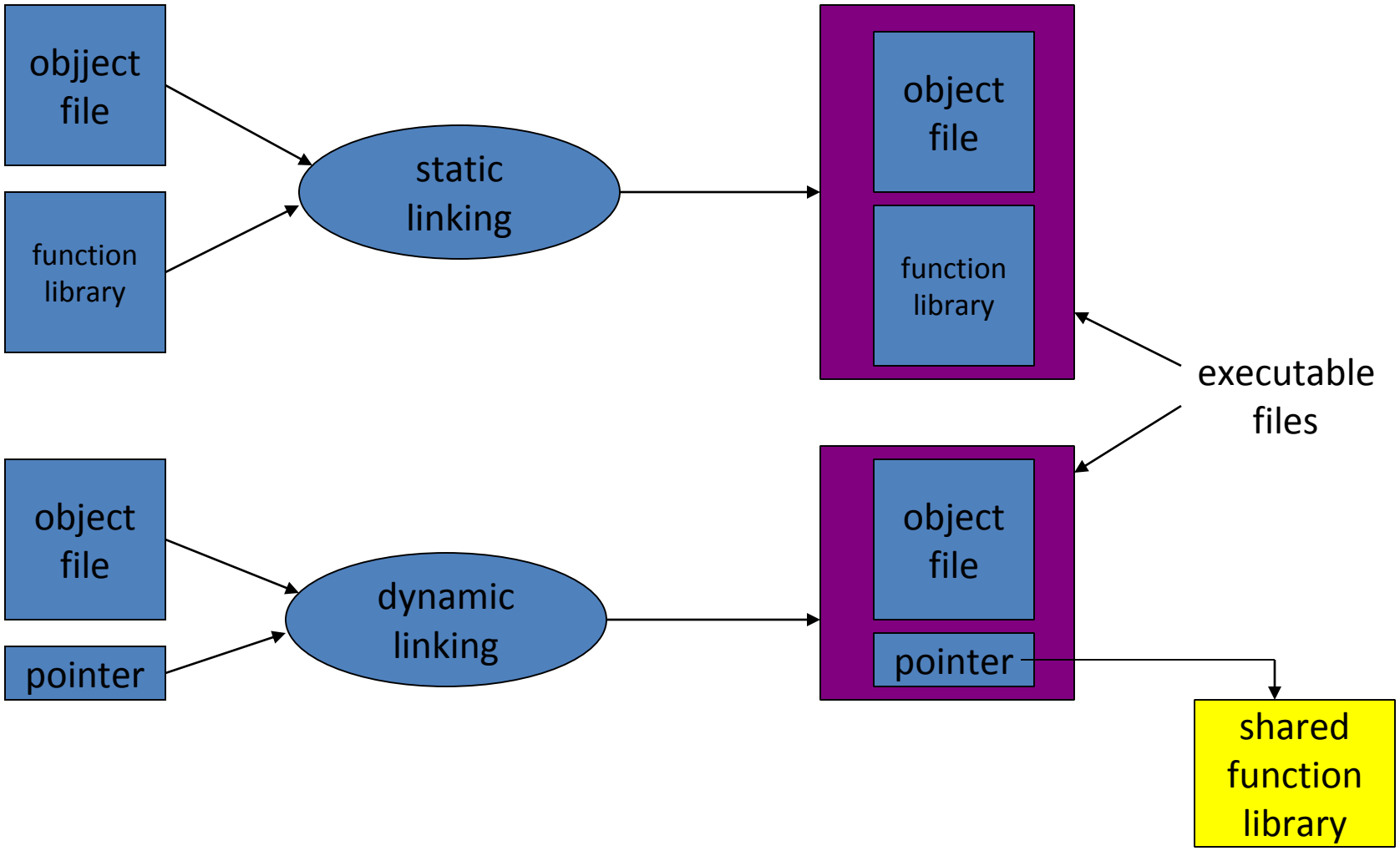
CS 35L

Spring 2020 – Section 7

Dynamic Linking

- Allows a process to add, remove, replace or relocate object modules during its execution.
- If shared libraries are called:
  - Only copy a little reference information when the executable file is created
  - Complete the linking during loading time or running time
- Dynamic libraries are typically denoted by the .so file extension
  - .dll on Windows

Smaller is more efficient



How are libraries dynamically linked?

Table 1. The Dl API

Function	Description
dlopen	Makes an object file accessible to a program
dlsym	Obtains the address of a symbol within a `dlopen`ed object file
dlerror	Returns a string error of the last error that occurred
dlclose	Closes an object file

Dynamic loading

```c
#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;
}
```

- Copy the code into main.c
- gcc main.c -o main -ldl
- You will have to set the environment variable LD_LIBRARY_PATH to include the path that contains libmymath.so

GCC Flags

- `-fPIC`: Compiler directive to output position independent code, a characteristic required by shared libraries.
- `-lXXX`: Link with "`libXXX.so`"
  - Without `-L` to directly specify the path, `/usr/lib` is used.
- `-L`: At **compile** time, find `.so` from this path.
- `-Wl,rpath=.`: `-Wl` passes options to linker. `-rpath` at **runtime** finds `.so` from this path.
- `-c`: Generate object code from c code.
- `-shared`: Produce a shared object which can then be linked with other objects to form an executable.

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)

Shared library name convention

- **soname:** has the prefix ``lib'', the name of the library, the phrase ``.so'', followed by a period and a version number that is incremented whenever the interface changes
  - (/usr/lib/libreadline.so.3)

- **Real name:** the filename containing the actual library code. The real name adds to the soname a period, a minor number, another period, and the release number. The last period and release number are optional
  - (/usr/lib/libreadline.so.3.0)

- **Linker name:** the soname without any version number
  - (/usr/lib/libreadline.so)

Names and Symbolic Links

- Linker name → soname
- /usr/lib/libreadline.so → /usr/lib/libreadline.so.3

- soname → real name
- /usr/lib/libreadline.so.3 → /usr/lib/libreadline.so.3.0

Attributes of Functions

- Used to declare certain things about functions called in your program
  - Help the compiler optimize calls and check code
- Also used to control memory placement, code generation options or call/return conventions within the function being annotated
- Introduced by the **attribute** keyword on a declaration, followed by an attribute specification inside double parentheses

Attributes of Functions

- `__attribute__ ((__constructor__))`
  - Is run when `dlopen()` is called
- `__attribute__ ((__destructor__))`
  - Is run when `dlclose()` is called
- Example:

```
__attribute__ ((__constructor__))
void to_run_before (void) {
    printf("pre_func\n");
}
```

Use the above in your implementation!