Quick notes

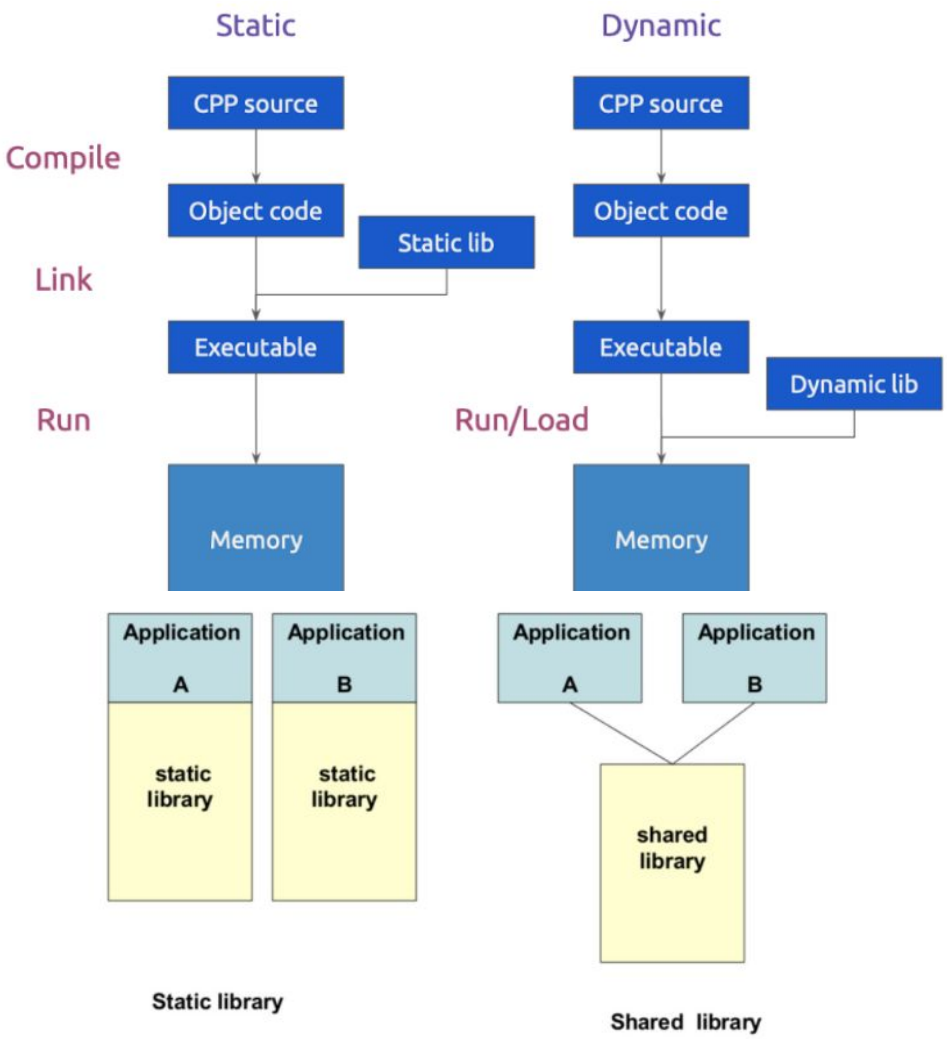
- Check Piazza
- Reminders:
  - Assign6 is due friday 20th Nov
  - Assign7 due 29th Nov
  - Weekly quiz
- Assignment 10 (this only applies to students enrolled in Lab 1)
  - **Email me your topic**
  - Create presentation (<10 min, semi-strict) + brief summary (<200 words)
    - Must present to class (email me if you're in a different timezone)
    - You can pre-record (e.g. youtube)
    - Starting next week!
  - Order is here, let me know ASAP if you are unable to present.
  - https://docs.google.com/spreadsheets/d/1rcPnnaa2zH8_3fi3qSLgCKRDdwKgqML32HocBnEv6q0/

Feedback / Office Hours / Other

- Tameez Latib
  - tameezlatib@gmail.com, please add “CS35L” to the subject line
  - Office Hours: Monday **4pm-6pm** (or by appointment)
  - Feedback: https://forms.gle/6kcJ2aJtzAzFMhHQ7 (anonymous google form)
- If you guys are stressed out:
  - CAPS (https://www.counseling.ucla.edu/)
    - Free with UC ship

Static vs dynamic linking

- Static
  - bigger files,
  - less runtime issues,
  - recompile if lib changes
- Dynamic
  - smaller files,
  - no recompile if lib changes,
  - slower runtime
- Demo
- (commands shown listed in last ppt)
- Ldd (with static vs dynamic linking)



Dynamic Loading

- Similar to dynamic linking
- Dynamic Loading: map/copy executable into memory **while process running**
  - Manual process, e.g. dlopen
- Dynamic Linking: Resolve symbols after compile time, **before process starts!**
  - OS does this for us

Dynamic Loading - how?

- Dlopen: open object file, e.g. dlopen(lib.so, flags)
- Dlsym: get symbol from opened object file
- Dlerror: return last error
- Dlclose: close
- Reference? Man pages, e.g.
  - https://www.man7.org/linux/man-pages/man3/dlopen.3.html

dlopen

- void *dlopen(const char *filename, int flags);
- General note about flags, it's a one-hot encoding
  - Each flag is actually an int, e.g. RTLD_LAZY = 1
  - To use multiple flags, OR them together
  - E.g. flag1 = 1 (bin 001), flag2 = 4 (bin 100)
    - Flag1 OR flag2 = 5 (bin 101)
- Here, RTLD_LAZY means that we only resolve symbols when they are needed; so if our code doesn't use a symbol it never loads

```
void * dl_handle;
dl_handle = dlopen("libglobal.so", RTLD_LAZY);
```

dlsym

- void *dlsym(void *handle, const char *symbol);
- Note the return type is void*
  - Returns a pointer to the symbol, so we must take care of type casting and de-referencing
- General note about typecasting
  - Be careful, sometimes things will automatically be typecast if you do not specify
  - This can cause issues
    - What if the data is uint, or int, or float?
    - E.g.

```c
int x = -1;
printf("x %d, x %u\n", x, x);
```

```c
int x;
x = *(int*) dlsym(dl_handle, "x");
```

dlerror

- char *dlerror(void);
- Returns error status of last dl command.
- If nothing went wrong, NULL

```
char * err = dlerror();
if (err != NULL){
        printf("There was an error: %s\n", err);
}
```

dlclose

- int dlclose(void *handle);
- Same as files (close), pointers (free), etc.
  - Technically dl_handles should be closed on exit, but it's good practice

```
dlclose(dl_handle);
```

Other dl stuff

- #include <dlfcn.h>
- Gcc main.c -ldl [-Wl,-rpath,.]
  - -ldl is needed to know dynamic loading will take place
  - -Wl,-rpath,path specifies a path to where the .so files are located, now can load with dlopen
- __attribute__((__keyword__))
  - Keyword can be constructor / destructor / etc
  - Constructor -> this fcn will be run on dlopen
  - Destructor -> this fcn will be run on dlclose
- https://gcc.gnu.org/onlinedocs/gcc-3.2/gcc/Function-Attributes.html

```
void on_open(void) __attribute__((__constructor__));
void on_open(){
    printf("Opened libglobal library");
}
```

Questions??