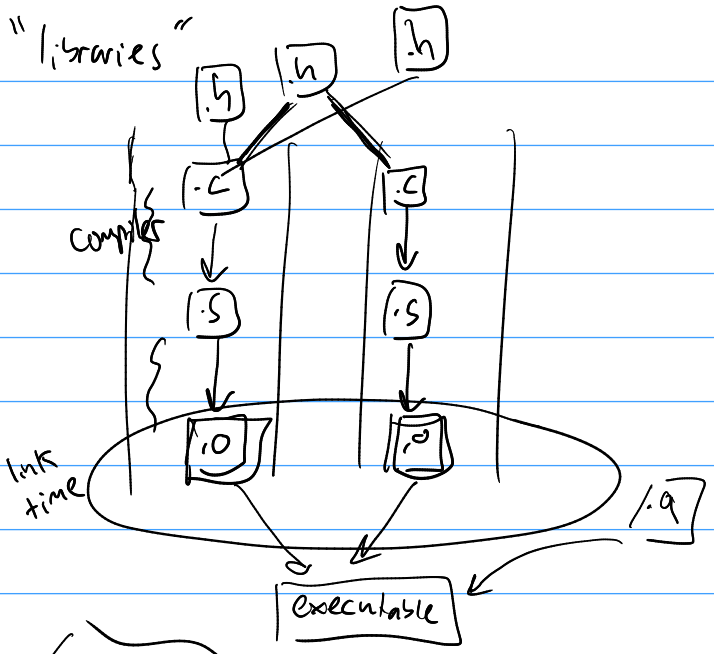
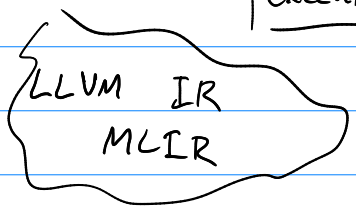
CS 3214 lecture #11 "libraries"

- static & dynamic libraries
- inlining

```
static inline int foo(int x) {
    ...
}
```

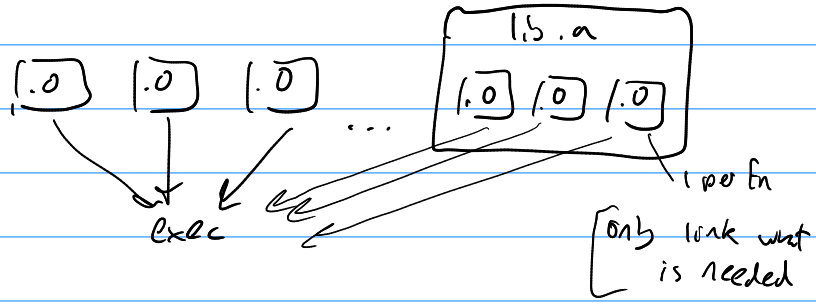


LTO: link time optimization



all .c
```
#include "foo.c"
#include "bar.c"
```
| "unity builds"

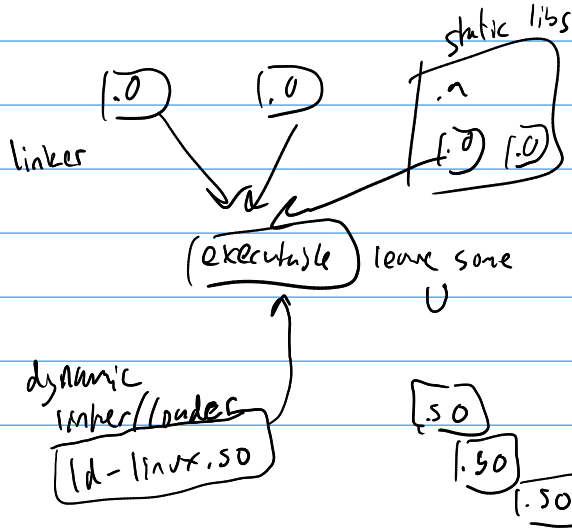
Libraries: static

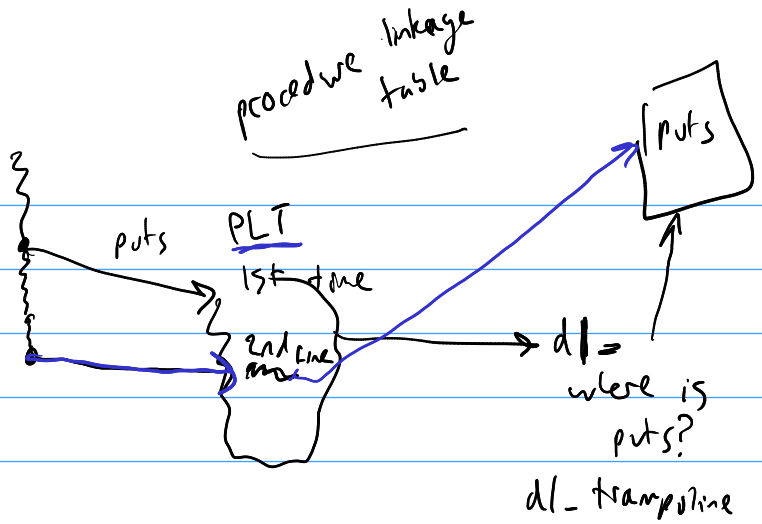


Linker goes through static libs
in order once, only picks up .o's for undef. symbols

{ + smaller ~~binaries~~ executable
- order matters

Dynamic libraries
.so "shared object"
.dll "dynamic link library"





Dynamic linking has other uses

LD_PRELOAD

Dynamic vs static

+ save mem
+ update of libs
  - sometimes they break things

+ safer... know everything is defined
+ organized, only what you need
+ faster

cow

Containers