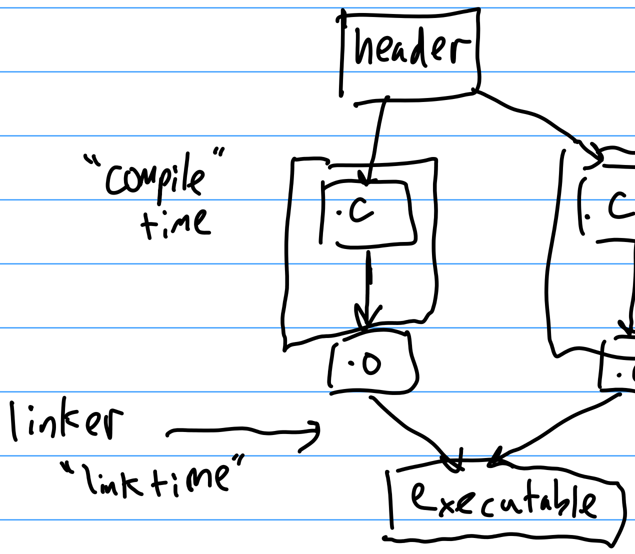
lecture #12

CS 3214 "linking libraries"

project 1
ex 2

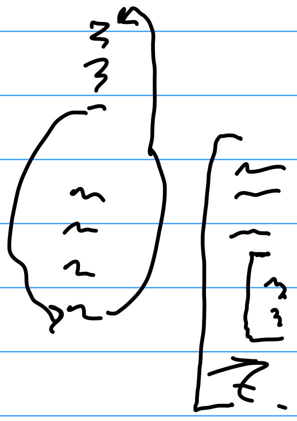
Next class 3/1 might be 5-10 late



linker:
resolving global symbols

declarations
definitions in C
- weak vs. strong

Best practices
- always use static when you can
- avoid global variables if possible
- do not <u>define</u> variables in header files
    - extern int foo; <u>one</u> hdr file
      int <u>foo</u>; only one c file
      list.c int list_foo;
- prototype function def hdr file



Inlining : compiler decision

header files { static inline foo {
    ....
  }

gcc -O0
gcc -O2

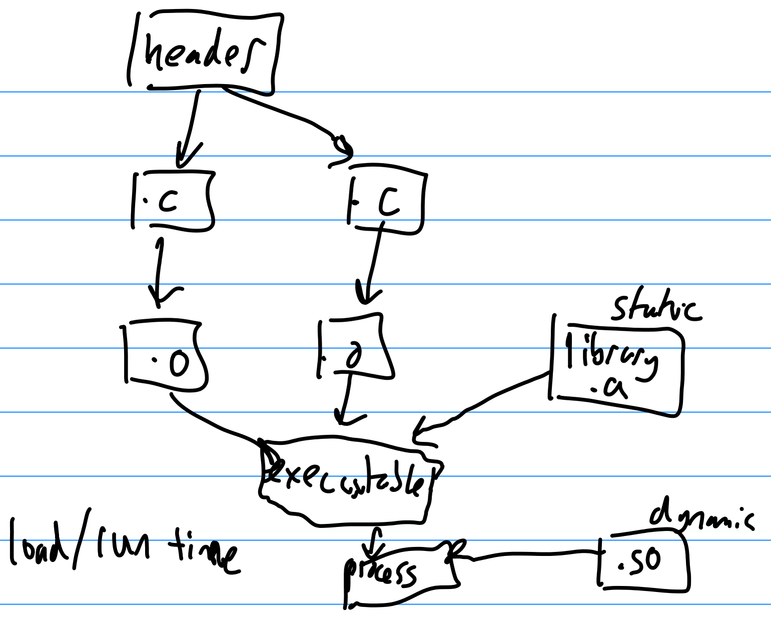
Link time Optimization LTO LLVM IR

libraries

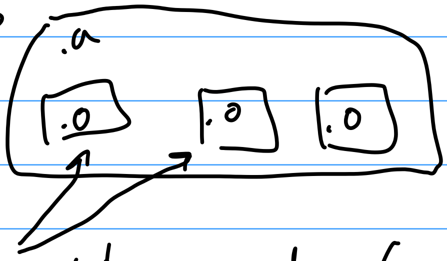
static .a

dynamic .so

link time



static libraries



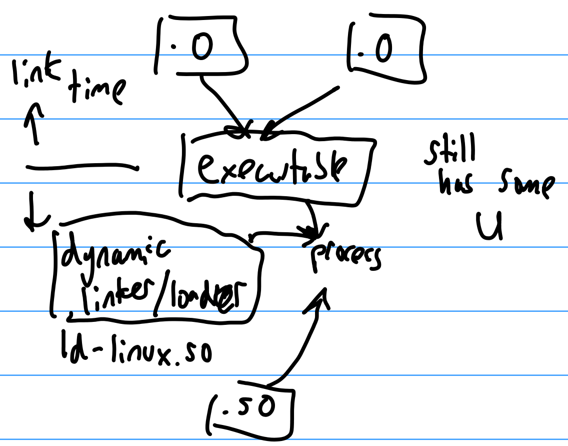
(small binaries)

→ linker maintains list of Undefined symbols
looks through libraries in order

musl libc

- every binary has a copy of library code
- update to library ⇒ recompilation
+ all dependencies are in binary

Alternative: Dynamic libraries .so
- shared object .so DLL
- loaded dynamically at runtime

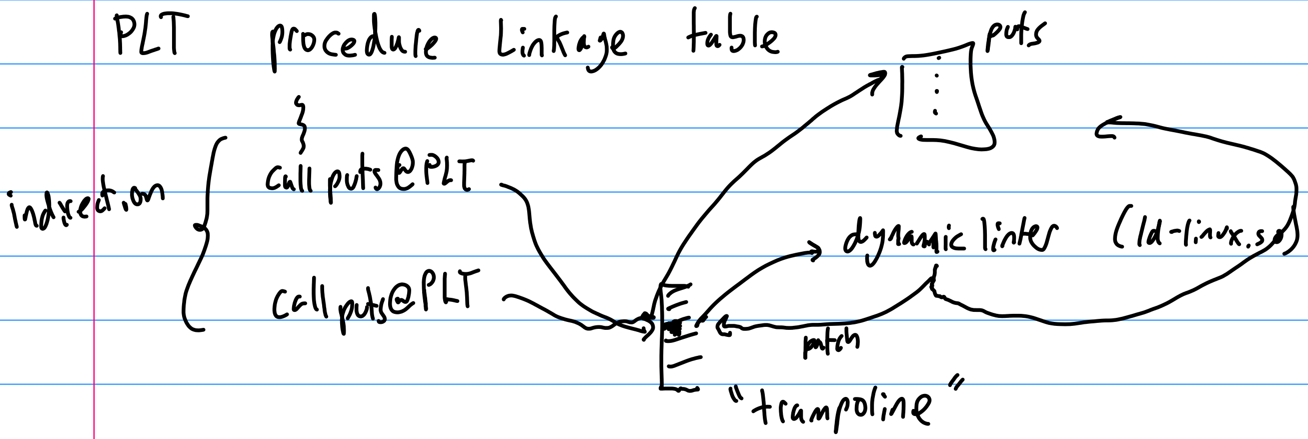


recursive
- .so may have deps

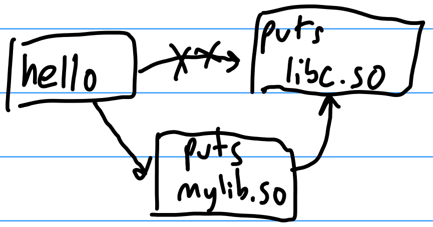
dlopen()

memory is shared between multiple processes

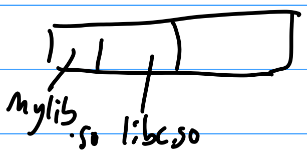
PLT procedure Linkage table



LD_PRELOAD



hello
puts
libc.so
puts
mylib.so



mylib.so
libc.so