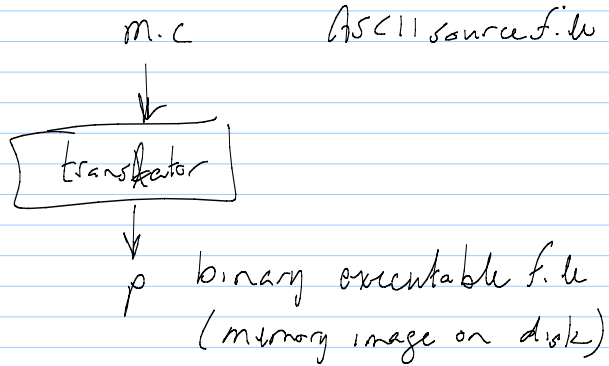


# Linking

Note Title

10/29/2007

Simple translation will work

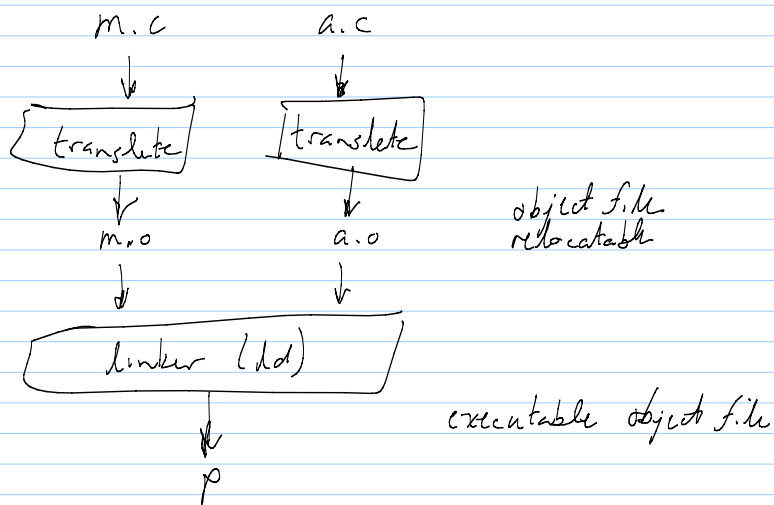


problems

- efficiency small change to source req. complete recompilation
- modularity hard to share common functions (e.g. printf)

solution

linker (static linker)



resolves external references

symbols declared in 1 file, used elsewhere  
relocate symbols from relative loc. in .o file to new absolute positions

update all references to relocated symbols

can be code: a) //symbol a

data: int \*xp = &x; //symbol x

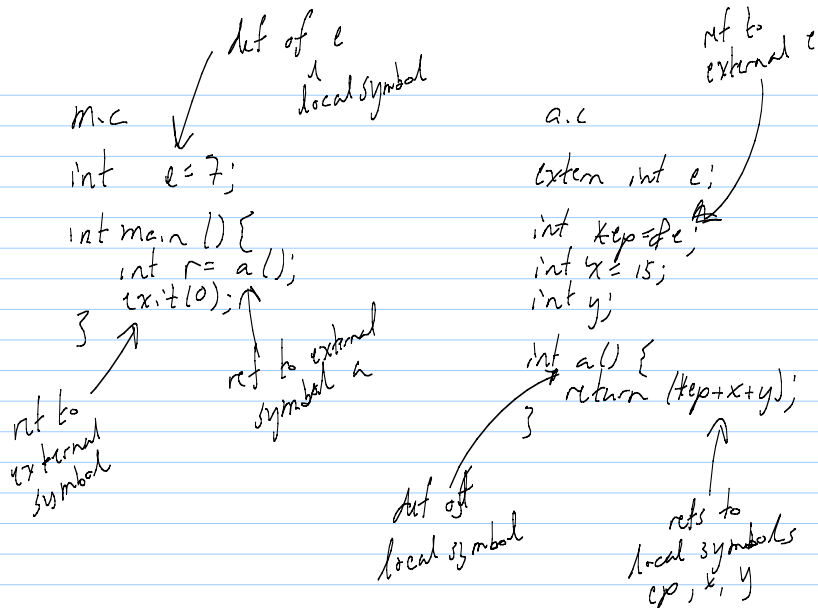
## Executable and Linkable Format (ELF)

std. format for many Unix-style OS

- o relocatable obj files
- executable binaries
- so shared object files

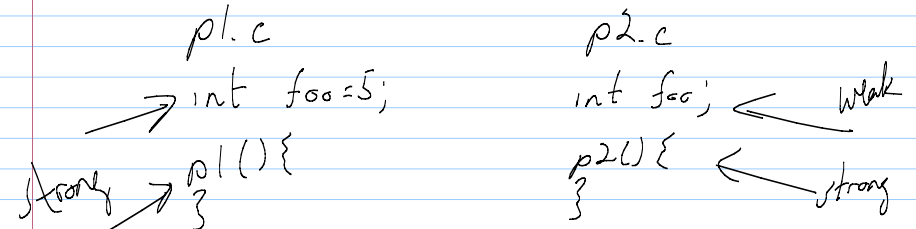
## Contents of ELF file:

ELF header	magic number, file type, machine
prog. header	virtual mem. addr., segments
• text section	executable code
• data section	initialized data
• bss section	uninit. data
• symtab section	global symbols
• rel. txt "	relocation info for code
• rel. data "	" " " data
• debug "	debugging symbol table (-g)
• line "	map line #s to machine code
• strtab "	constant strings table
section header table	location of sections within ELF



program symbols are "strong" or "weak"

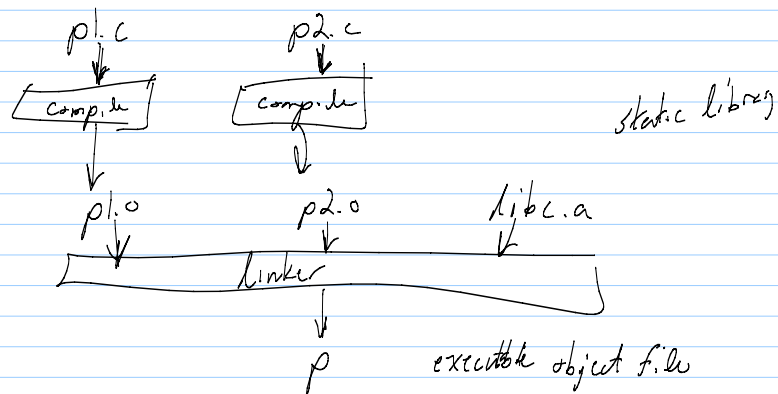
- strong: procedure names and initialized global variables
- weak: uninitialized global variable



## Linking rules

1. A strong symbol can appear only once
2. A weak symbol can be overridden by a strong symbol of the same name  
- refs to weak symbol resolve to strong sym.
3. If there are multiple weak symbols, the linker can pick arbitrary one.

<code>int x;</code>	<code>p1() {}</code>	<code>p1() {}</code>	error!
<code>int x;</code>	<code>p1() {}</code>	<code>int x;</code>	<code>p2() {}</code>
<code>int x;</code>	<code>int y;</code>	<code>double x;</code>	<code>p2() {}</code>
<code>int x; 7</code>	<code>int y; 7</code>	<code>double x;</code>	<code>p2() {}</code>
<code>p1() {}</code>			Err!.



includes only function in libc that are ref. by p1, p2

