Linking

Simple translation will work

```
.m.c  ASCII source file

```

Translator

```
\[\rightarrow\]  binary executable file

(memory image on disk)
```

```
\[\rightarrow\]  a.o  object file

(relocatable)

```

```
\[\rightarrow\]  m.o  object file

```

```
\[\rightarrow\]  a.o  object file

```

```
\[\rightarrow\]  bin (ld)

(linker)

```

```
\[\rightarrow\]  executable object file

```

- efficiency: small input vs. large output
- modularity: hard to share common code
  functions (e.g., printf)
  symbols
  linker (static linker)

resolves external references

symbols declared in .o file, used elsewhere
relocate symbols from relative .o in .o
file to new absolute positions
update all references to relocated symbols

```
\[\rightarrow\]  a1) \#1 symbol

```

```
\[\rightarrow\]  a2) \#2 symbol

```

```
\[\rightarrow\]  a3) \#3 symbol

```

```
\[\rightarrow\]  a4) \#4 symbol

```

```
\[\rightarrow\]  a5) \#5 symbol

```

```
Executable and Linkable Format (ELF)

- std. format for many Unix-style OS
  - 2 relocatable obj. files
  - 1 executable binary
  - shared object files

Contents of ELF file:

- ELF header
  - magic number, file type, version
- program header
  - program header table
  - section header table
- section header
  - section name, string table
  - symbol table
  - global symbols
  - relocation info for code
  - relocation info for data
  - program header (e.g.)
  - mapping of data to memory and code
  - section header: location of sections within ELF

Program symbols are "strong" or "weak":

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

1.c

```c
int x = 5;
int y;
int main()
```

2.c

```c
int foo = 5;
int bar;
int baz = 3;
```
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 any one.

```
int x;
int *x;
p1() -> p1() x3  p2() x3  error!
p1() -> p1() x3
int x;
p1() -> p1() x3  p2() x3
int x;
double x;
p1() -> p1() x3
int x2;
double x;
p1() -> p2() x3
int x2;
double x;
p1() -> p2() x3  err:1
```

```
p1.c    p2.c
  ↓     ↓
  compile  compile
  ↓     ↓
  p1.o    p2.o
  ↓     ↓
  linker

linker

p  executable object file
  include only function in libc that are ref by p1, p2
```

```
m.c    a.c
  ↓     ↓
  m.o    a.o
  ↓     ↓
  sharedlib
  ↓
  libc.so
  ↓
  header/dynamic linker
  ↓
  ld-linux.so
  ↓
  fully linked executable
  ↓
```