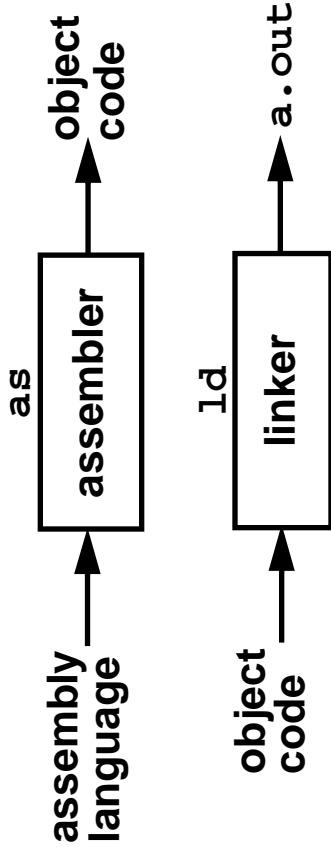


Linking and Loading

- Generating an a.out is a two-step process



- Why two steps?
- Linkers combine multiple object files into one executable file
e.g., a.out, and perform external symbol resolution and relocation
- Other linker capabilities: segments, e.g., text & data, library searching...

Example

- **main.c:**

```
static int a;
extern int b;
int c[10];

main() {
    f(a, b);
    g(a, b);
}
```

- **f.c:**

```
int b;

f(int a, int b) {
    return a + b;
}

g(int a, int b) {
    return a - b;
}
```

- **Compiling and linking**

```
% lcc -v -c main.c
/usr/princeton/lib/gnu/gcc-cpp ... main.c |
  /usr/princeton/lib/rcc -v -o /tmp/lcc24417.s
/bin/as -o main.o /tmp/lcc24417.s
rm /tmp/lcc24417.s
% nm main.o
00000050 b _a
          U _b
00000028 C _c
          U _f
          U _g
00000000 T _main
%
% lcc -c f.c
% nm f.o
00000004 C _b
00000000 T _f
0000000c T _g
% lcc -v main.o f.o
/bin/ld -o a.out -dc -dp
-e start -X /usr/lib/crt0.o
main.o f.o -lm -lc
```

Relocation

- **Relocation**: assembler assumes each module begins at address 0
- When `main.o` and `f.o` are linked, addresses must be **relocated**, why?
- Modules are concatenated in the order in which they are given to `ld`;

but the compiler might **rearrange** data or functions arbitrarily

```
% lcc main.o f.o
% nm a.out
00004000 d __DYNAMIC
000040c8 b a
00004098 B b
000040a0 B c
00004098 D _edata
000040d0 B _end
00004090 D _environ
000024c4 T _etext
000022e0 T f
000022ec T g
00002290 T main
...
```

```
% lcc f.o main.o
% nm a.out
00004000 d __DYNAMIC
000040c8 b a
00004098 B b
000040a0 B c
00004098 D _edata
000040d0 B _end
00004090 D _environ
000024c4 T _etext
00002290 T f
0000229c T g
000022a8 T main
...
```

- Object code contains information about **relocation sites**

Symbol Resolution

- Linking must **resolve all symbols**; undefined symbols are errors

```
% lcc main.o
ld: Undefined symbol
   _b
   _f
   _g
```

- Linking must **define each symbol once**; multiple definitions are errors

```
% cp f.o g.o
% lcc main.o f.o g.o
ld: g.o: _f: multiply defined
   g.o: _g: multiply defined
```

- Linkers **do not type check** across modules; types of undefined and defined symbols must agree
- Linkers combine all modules specified, even if some are not referenced
- Undefined symbols drive **library searching**

```
lcc -o cw main.o /u/cs217/lib/lib217.a -lc
```

main.o references but does not define **parse**, **Table_***, and C library functions only those modules that define one of these symbols are linked from **lib217.a** and from the C library (specified by **-lc**)

Linking and Loading, cont'd

- “Linking loaders” consume object files, produce an executable file
- “Link editors” consume object files, produce executable *or object file*
- Link editors (like `ld`) can build an object file iteratively


```

% ld -o ab.o -r a.o b.o
% ld -o a.out ab.o c.o
-r causes ld to keep relocation information and generate an object file
executable files normally have no relocation information

```
- Executable files do contain a *symbol table*, which is useful for debugging but takes space; it can be removed by “stripping” the executable

```

% lcc -o cw main.o lib217.a
% ls -l cw
24576 cw
% nm cw
000028a0 T _Table_install
...
00002d88 T _parse
...
% strip cw
% ls -l cw
16384 cw
% nm cw
nm: cw: no name list

```