Final Remarks on Loaders/Linkers

Linking C programs with libraries:

- Functions from <stdlib.h> (e.g. malloc, free, exit) are linked automatically.
- Functions from <stdio.h> (e.g. fprintf, fscanf) are linked if <stdio.h> is included as a header.
- If functions from other libraries (e.g. <math.h>) are used, then the corresponding header must be included and the library must be specified as part of the gcc command.

Example: Suppose a program (say prog.c) uses the function sqrt from the math library.

(a) The program must include <math.h>.

(b) Command for producing load module:

```
% gcc  prog.c  -lm
```


Notes:

- The linker searches the libraries only after obtaining information about all the functions defined in the various files.
- This allows a user to redefine a library function. (If this is really necessary, it must be done with great care.)

Dynamic Linking:

- Static linking: Done before execution.
- Dynamic linking: Done during execution.
- Dynamic linking is useful when a program calls only a few of a large collection of routines from a library (e.g. a library containing error handling routines).
- Advantage: The size of the load module is smaller.
- Disadvantage: Runtime overhead.

Steps used in Dynamic Linking:

1. A user program calls a routine (say `sqrt`) which is not part of the load module.
2. The runtime system makes a service request to the OS to load the requested function. The request is handled by a part of the OS called the "Dynamic Loader" (DL).
3. DL checks if the requested routine is already in memory; if not, loads the routine at an appropriate part of memory. DL also starts the execution of the routine.
4. When the routine completes, it returns control to DL. DL decides whether to leave the routine in memory or reclaim the memory.
5. DL passes the control back to the user program.

Notes:

- Binding: Association of an attribute value with a name.
- Binding can happen at compile time, at load time or at execution time.
- With static linking, the binding between a function name and its address happens at load time (before execution begins). For a given execution, this binding does not change with time (static binding).
- With dynamic linking, the binding between a function name and its address may happen at execution time; this binding may also change with time (dynamic binding).

Bootstrap Loader:

Issue: How to load the very first program into memory.

- A small part of memory is implemented as Read Only Memory (ROM) which contains a program. (This program is an absolute loader.)
- When power is turned on (or when the system is rebooted), the ROM program begins to execute. The program reads a block of bytes from a specific device (e.g. hard disk, floppy disk).
- The block (called the "boot block") read in by the ROM program contains another program which can load a larger program.
- Thus, the program in the boot block loads a larger program, ..., and so on, until the OS itself is loaded.
- This sequence of loading larger and larger programs until the OS is loaded is called **bootstrapping**.
- Systems also perform "Power on self-tests" during booting.

Note: Some people refer to the ROM program as the bootstrap loader; others refer to the program in the boot block as the bootstrap loader.

Reading Assignment: Figure 3.3 of [Beck].

Notes about the program in Figure 3.3:

- It is a bootstrap loader (i.e., a simple absolute loader) for SIC/XE (that can be stored in ROM).
- It reads bytes from device F1 (hex) and loads the bytes from address 80 (hex).
- The device returns 04 (hex) when EOF occurs.
- At that time, the program causes a jump to location 80 (hex) to start the program that was just loaded.