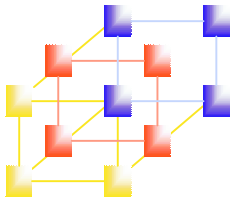
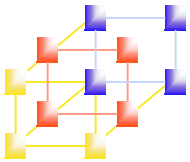


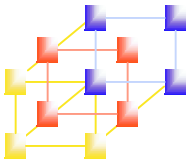
Chapter 3 Loaders and Linkers

-- Loader Design Options



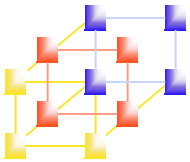
Loaders

- Linkage editor
  - Linking before loading
- Dynamic linking
  - Linking at the execution time
- Bootstrap loader



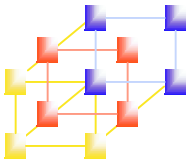
Linkage editors

- Difference between a linkage editor and a linking loader:
  - Linking loader
    - Performs all *linking* and *relocation* operations, including *automatic library search*, and *loads* the linked program into memory for execution.
  - Linkage editor
    - *Produces a linked version* of the program, which is normally written to a file or library for later execution.
      - *A simple relocating loader* (one pass) can be used to load the program into memory for execution.
      - The linkage editor performs relocation of all control sections relative to the start of the linked program.
      - The only object code modification necessary is the addition of an actual load address to relative values within the program



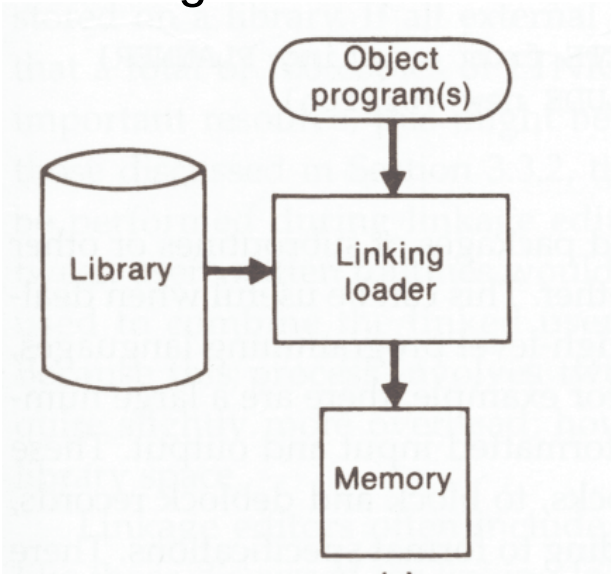
Compare between linking loader and linkage editor

- Linking loader
  - Suitable when a program is reassembled for nearly every execution
    - `In a program development and testing environment`
    - `When a program is used so infrequently that it is not worthwhile to store the assembled and linked version.`
- Linkage editor
  - Suitable when a program is to be executed many times without being reassembled because resolution of external references and library searching are only performed once.

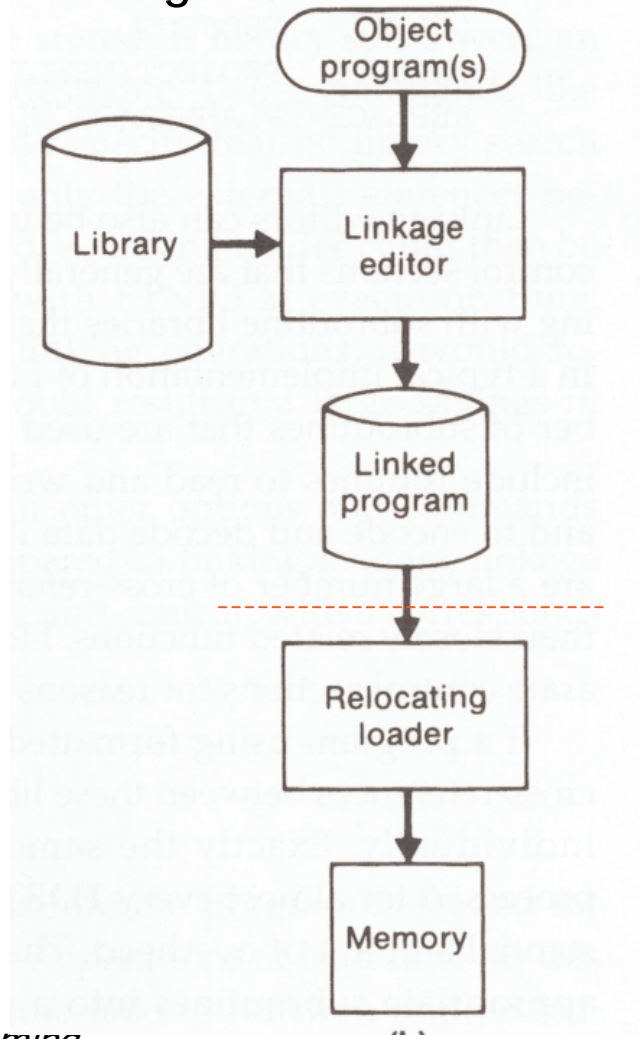


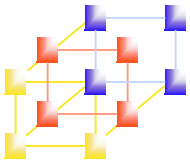
Linking loader *v.s.* linkage editor

Linking loader



Linkage editor





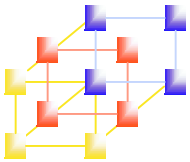
Additional Functions of Linkage Editors

- Replacement of subroutines in the linked program
  - For example:

```
INCLUDE    PLANNER(PROGLIB)
DELETE     PROJECT    {DELETE from existing PLANNER}
INCLUDE    PROJECT(NEWLIB)  {INCLUDE new version}
REPLACE    PLANNER(PROGLIB)
```

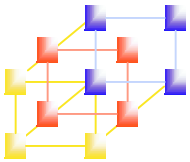
- Construction of a package for subroutines generally used together
  - There are a large number of cross-references between these subroutines due to their closely related functions.
  - For example (P.155) :

```
INCLUDE     READR(FTNLIB)
INCLUDE     WRITER(FTNLIB)
       :
SAVE        FTNIO(SUBLIB)
```



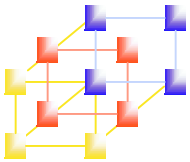
Additional Functions of Linkage Editors (Cond.)

- Specification of external references not to be resolved by automatic library search
  - Can avoid multiple storage of common libraries in programs.
  - Need a linking loader to combine the common libraries at execution time.



Dynamic Linking

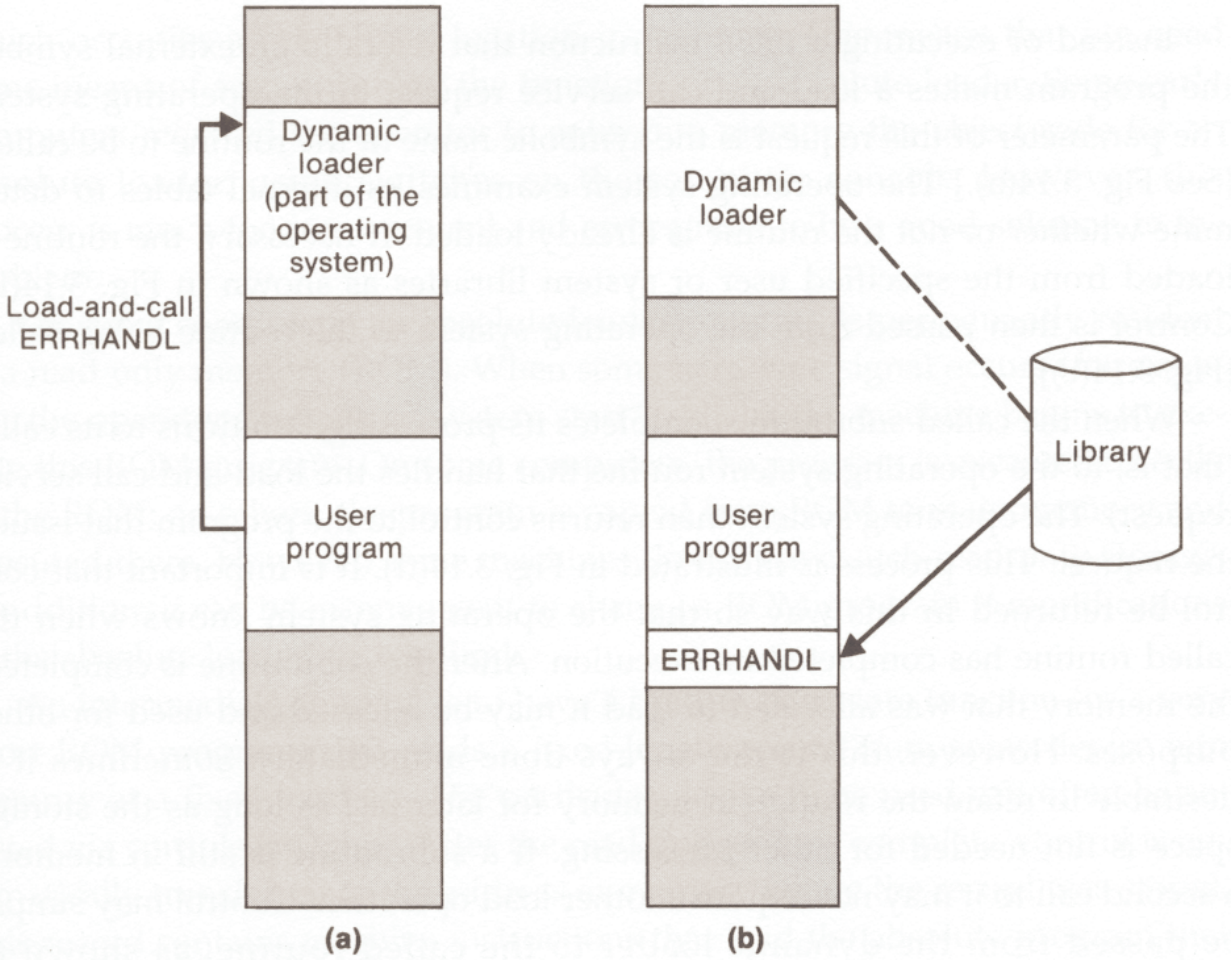
- Comparison
  - Linkage editors perform linking operations *before* the program is loaded for execution
  - Linking loaders perform linking operations *at load time*
  - Dynamic linking (dynamic loading, load on call) perform linking *at execution time*
- Delayed Binding
  - Avoid the necessity of loading the entire library for each execution, i.e. load the routines only when they are needed
  - Allow several executing programs to share one copy of a subroutine or library (Dynamic Link Library, DLL)



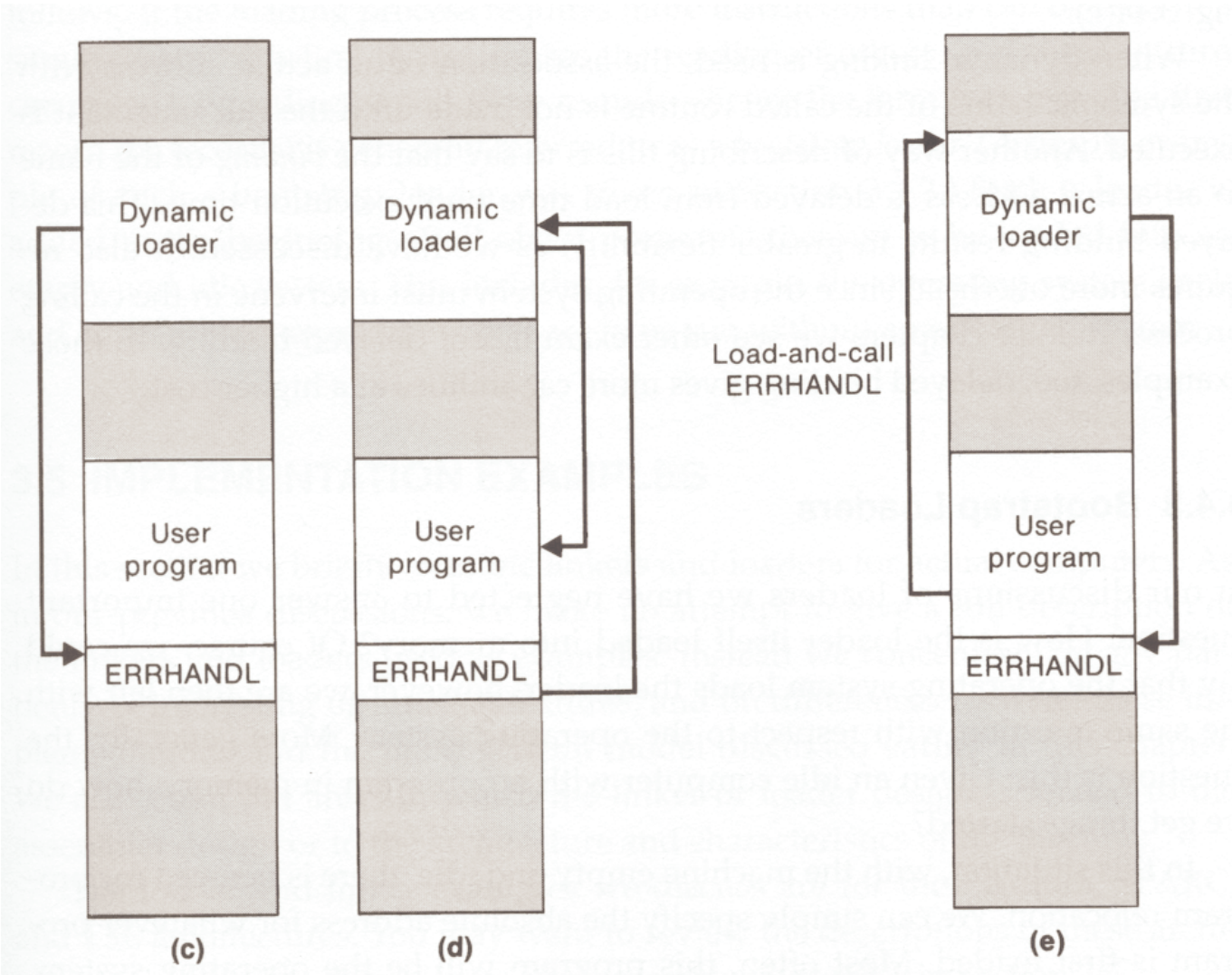
Dynamic Linking

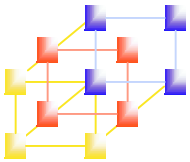
- O.S. services request of dynamic linking
  - Dynamic loader is one part of the OS
  - Instead of executing a JSUB instruction that refers to an external symbol, the program makes a load-and-call service request to the OS
- Example (Figure 3.14)
  - When call a routine, pass routine name as parameter to O.S. (a)
  - If routine is not loaded, O.S. loads it from library and pass the control to the routine (b and c)
  - When the called routine completes it processing, it returns to the caller (O.S.) (d)
  - When call a routine and the routine is still in memory, O.S. simply passes the control to the routine (e)

Example of Dynamic Linking -- Figure 3.14, pp.157



Example of Dynamic Linking -- Figure 3.14, pp.157





Bootstrap Loaders

- How is the loader itself loaded into memory?
  - An absolute loader program is permanently resident in a read-only memory ROM
  - Copy absolute loader in ROM into RAM for execution (optional)
  - Read a fixed-length record from some device into memory at a fixed location. After the read operation, control is automatically transferred to the address in memory