

Program Loading



- **Assume virtual memory**
 - Each process has own VAS
- **But program has components**
 - Object files, libraries, etc.
- **Combine into one executable**
 - Allocate virtual address space

Object Files

- **Produced by compiler**
 - Instructions and data
 - Correct virtual addresses unknown
- **Combined by linker (UNIX "ld")**
 - Single binary file
 - Ready for loading into memory

Relocation

- **Compiler address allocation**
 - Typically starting at address zero
 - Multiple segments?
 - Code, data, stack
- **Translated to actual virtual**
 - Address modification
 - Instruction offsets, pointers, etc.

Global References



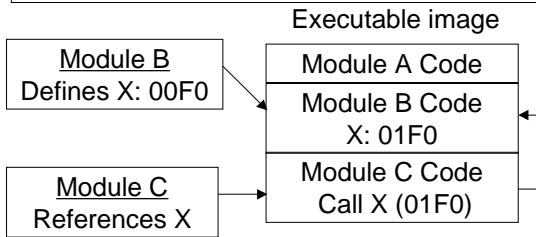
• Global symbols

- Defined in one object module
- Referenced in another

• Resolved by linker

- Global symbol table
 - In object module
- Dummy addresses replaced with real

Global Reference Resolution



Library References

• Linker combines object modules

- Resolves global symbols
- Some references not satisfied

• Search specified libraries

- Object module collections
- Add modules with needed globals

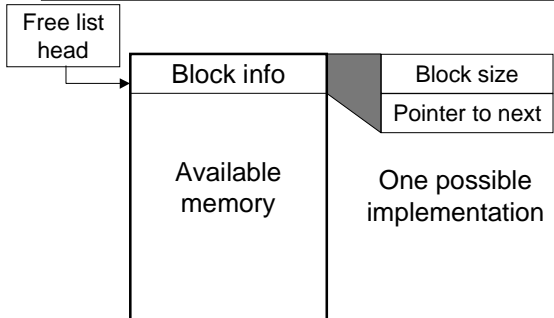
Shared Libraries

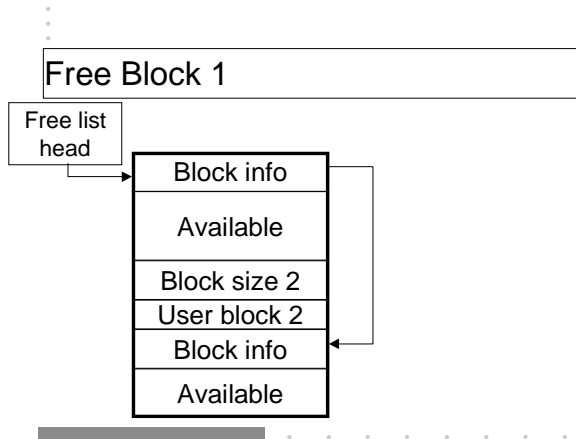
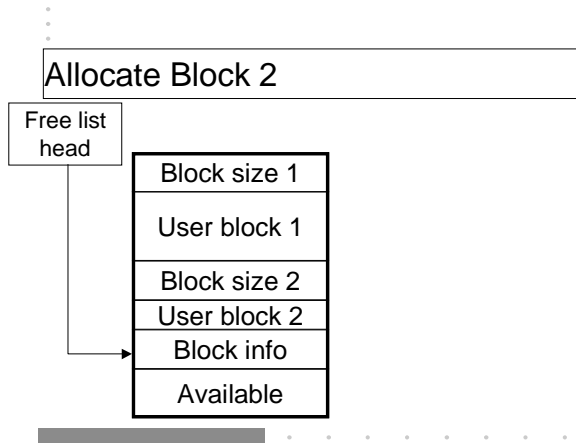
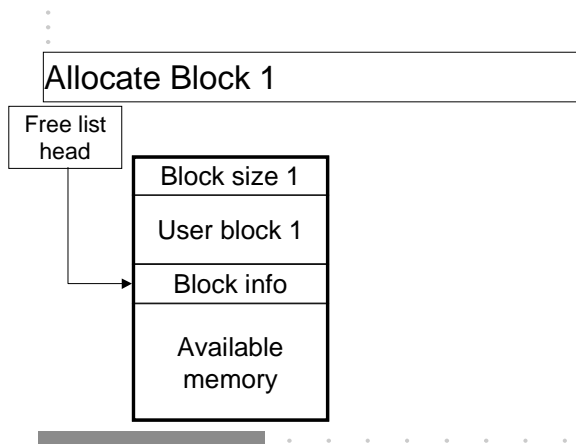
- **Normal library code** (e.g., `libc.a`)
 - Linked into each executable
- **Shared library code**
 - Pre-linked into memory segment
 - Fixed relative entry points?
 - Executable references satisfied

Dynamic Memory

- **Allocated as needed**
 - As program executes
 - E.g., `new`, `delete`
- **Source of memory**
 - Segment within program
 - Expandable at run time?
 - Managed by library code

Initialize Dynamic Memory

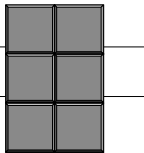




Dynamic Allocation

- **Maintain list of free blocks**
 - Size, pointer to next block
- **Allocate block as needed**
 - Round up to standard size?
 - To avoid fragmentation
 - May mask some allocation errors!!
- **Return to list on deallocation**

Allocation Algorithm



- **First fit**
 - Find first block that is big enough
 - Allocate block to user
 - Whole block, or split into used/free
- **Best fit**
 - Search for smallest "big enough"

Allocation Failure

- **What if no block found**
 - No free blocks
 - Largest block not big enough
- **Give up**
 - Return NULL from new
 - Throw exception
- **Ask OS for more**
 - Increase size of heap segment

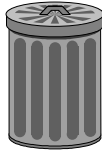
Memory Leaks

- **Memory allocated by program**
 - No longer in “free” list
- **Program fails to free**
 - Not available for reallocation
- **Allocation failure**
 - Or heap segment grows to limit



Garbage Collection (1)

- **Detect end of use**
 - Know when program no longer needs memory block?
 - Free block automatically
- **Explicit tracking**
 - Reference count
 - # of pointers to block



Garbage Collection (2)

- **Little support in C or C++**
 - Some other languages do better
 - LISP, Java
- **Best guess?**
 - Search data space for pointers
 - But how do you know it's a pointer?
- **Circular reference problem**
