UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 537
Introduction to Operating Systems

Andrea C. Arpaci-Dusseau
Remzi H. Arpaci-Dusseau

Haryadi S. Gunawi

Dynamic Memory Allocation

Virtual Memory Mgmt. Summary

Hardware/physical memory:

- Can be 1-GB, 2-GB, 10-GB, 128-MB
- Users/programmers do not need to care about low-level details → the need for Virtual Memory Management in OS

A user's view of memory is the *virtual address space*

- How big? Depend on the arch: 32 bit = 2^32 addressable bytes = 4GBytes
- If you need more? Special handling (compilation)

Summary: 2-levels of memory management

- The OS manages the physical memory
- The malloc/free library manages the allocation/deallocation in virtual address space

Mem. Mgmt. High Level

You have a chunk of memory to use

- (e.g. 1000 bytes)

A user wants to:

- p4 = malloc(40), p3 = malloc(30), p2 = malloc(20), p1 = malloc(10)

Then the user wants to:

- free(p4), free(p2)
- How does the lib know that p4 is 40 bytes, and p2 is 20 bytes??

Then the user wants to:

- malloc(5)
- Which holes do you give? Depend on the policies

So, the job of a malloc library:

- Manage allocated/free space
- Among all available free blocks, which one to give?

Heap Allocation Policies

Best fit

- Search entire list for each allocation
- Choose free block that most closely matches size of request
- Optimization: Stop searching if see exact match

First fit

- Allocate first block that is large enough

Rotating first fit (or "Next fit"):

- Variant of first fit, remember place in list
- Start with next free block each time

Worst fit

- Allocate largest block to request (most leftover space)

Heap Allocation Examples

Scenario: Two free blocks of size 20 and 15 bytes

Allocation stream: 10, 20

- Best: 20 5 | 5 (win!)
- First: 10 15 (fail)
- Worst: 10 15 (fail)

Allocation stream: 8, 12, 12

- Best: 20 7 | 8 7 (fail)
- First: 12 15 | 15 | 3 (win!)
- Worst: 12 15 | 15 (fail)

Comparison of Allocation Strategies

No optimal algorithm

- Fragmentation highly dependent on workload

Best fit

- Tends to leave some very large holes and some very small holes
  - Can't use very small holes easily

First fit

- Tends to leave "average" sized holes
- Advantage: Faster than best fit
- Next fit used often in practice

Memory Allocation in Practice

How is malloc() implemented?

Data structure: Free lists

- Header for each element of free list
  - pointer to next free block
  - size of block
  - magic number
- Where is header stored?
  - Read the C book (Kernighan), goto index pages, and search for malloc

Two free lists (advanced technique)

- One organized by size
  - Separate list for each popular, small size (e.g., 1 KB)
  - Allocation is fast, no external fragmentation
- Second is sorted by address
  - Use next fit to search appropriately
  - Free blocks shuffled between two lists

Heap Implementation

Data structure: **free list**

- Linked list of free blocks, tracks memory not in use
- Use the free space to store the link list
- **Header in each block**
  - size of block
  - ptr to next block in list

void *Allocate(x bytes)

- Choose block large enough for request (>= x bytes)
- Keep remainder of free block on free list
- Update list pointers and size variable
- Return pointer to allocated memory

Free(ptr)

- Add block back to free list
- Merge adjacent blocks in free list, update ptrs and size variables