ECE 531 – Advanced Operating Systems
Lecture 18

Vince Weaver

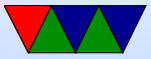
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

15 October 2025

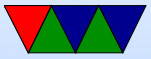
Announcements

- Midterm the 20th



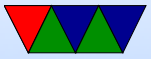
Project Preview

- Project pdf posted to course website
- Can work in groups
- First deadline is project topic, due November 5th
- Status update before Thanksgiving. Requires short literature search
- Project presentations last week of class
- Final writeup due last day of classes (December 19th)
- List of topics in the handout on website



Kernel Memory Handling

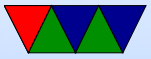
- The kernel also needs to manage memory
- Needs to allocate memory for processes and such
- Might need to dynamically allocate stuff in general operation though that's sometimes discouraged (local variables on stack much easier to deal with than fragmentation issues from malloc() like allocations)



Detecting Memory

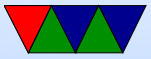
- How do we know how much memory we have?
  - Firmware – ask the bootloader/firmware (but how does it know?)
  - Assume – if only running on a machine with a fixed amount
  - Probing – try writing a value to memory, then read it back and see if it's the same

    Might read a bunch before writing to avoid false positives from bus capacitance



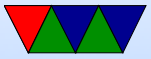
Tracking Memory – Granularity

- What granularity should be used?
- Too small (byte granularity) too much overhead to track it all
- Too large, then hand out much bigger chunks than actually need



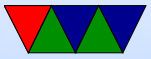
Tracking Memory – Data Structures

- There are various data structures used for tracking available memory



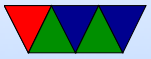
Tracking Memory – Bitmaps

- Bitmaps are the simplest
- Bitmap of chunks of memory, each bit indicated free/used
- Have to search bitmap and find N consecutive empty areas for each allocation
- Lots of bit-fiddling though some architectures have instructions (popcnt) that make this easier
- We'll see this is also used by some filesystems



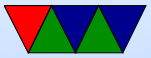
Bitmap Example

- 2GB of memory, memory broken up into 1k chunks
- 32-bit system so each word 4 bytes
- 2GB/1k = 2MB entries to track, each one a bit, so 2MB/32 = 64k of integers array (256k total size)
- To find if memory address is used/free, take address, /1024/32 to get index in array, then use bottom 5 bits to pick which bit to look at



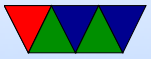
Bitmap — How Many Blocks/Chunks Needed?

- Want to allocate size 18? (1 block)
- Want to allocate size 1500? (2 blocks)
- Want to allocate size 6000? (6 blocks)
- Want to allocate size 8192? (8 blocks)
- ((size-1)/1024)+1 to find out how many blocks you need



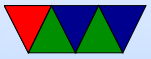
Tracking Memory – Other Data Structures

- Free-lists, linked list of memory areas
- Trees



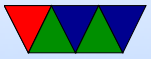
Fragmentation

- Enough memory available, but split up. How can fix?
- Memory compaction. Swap everything out, bring it back in (Relocating)
- Is this always possible? On Java? In C?



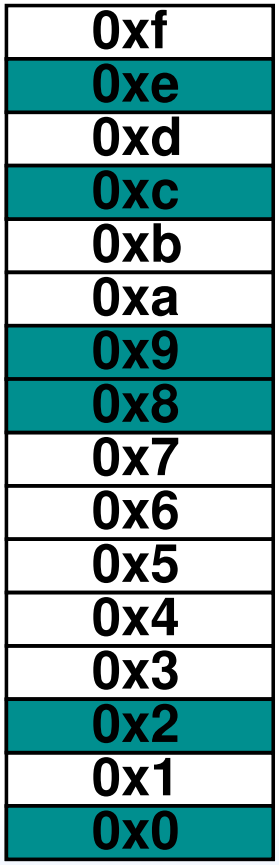
Fit Algorithms

- **First Fit**: scans bitmap, returns first block big enough to meet request. Fastest.
- **Next Fit**: Picks up where the last first fit case left off (optimization)
- **Best fit**: search entire map and find hole that fits it best. Actually can cause more fragmentation, end up with lots of tiny holes
- **Worst Fit**: always biggest hole. Not so great either.
- **Quick Fit**: separate lists for more common sizes



Fit Example

64kB memory



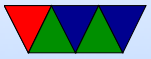
Memory Usage Bitmap

0101 0011 0000 0101

= Used Memory

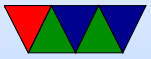
= Free Memory

Each page of memory is 4kB

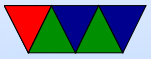


Fixed Sized Allocation

- Memory Pool – fast – blocks of pre-allocated memory in power of two sizes that can be handed out
  fast to allocate/free
  fragmentation

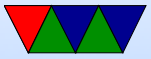


Brief History of Memory Handling in Operating Systems



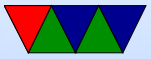
Mono-Programming

- Simple mono-programming: just OS and one program in memory at once (like DOS)
- Linear physical memory, assume you have all (or maybe up to a limit set by OS)
- Hassle of DOS 640k low memory, games



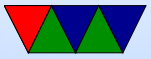
Fixed Multi-Programming

- Multiprogramming: let you run multiple tasks at once.
- Fixed Partitions of memory available. Jobs queued. When spot frees up job can run. Can have complex scheduling rules out which size and priority to give to jobs. Older mainframes (OS/MFT) used this.
- Relocations a problem
- Memory protection. Permissions on pages.
- Solution to both protection and permission in segments (with base offset and range that are valid to access)

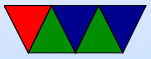


Swapping

- Timesharing systems. All jobs not fit in RAM?
- Swapping: bring in each program in entirety, run it a while, then when done writing all back out to disk.
- Paging: virtual memory.

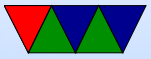


Memory Allocation in Linux



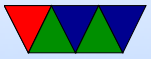
Buddy Allocator

- Used by Linux
- Pick a low size, say 4k, and a high size, say 1MB
- When allocate, round up to the next power of two
- Search for free area that size. If not, scale up. If you find one, split it into chunks until you reach the size being looked for. Give it.
- When freeing, not only free but see if neighboring blocks also free, if so, re-join them to bigger sized memory.



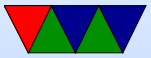
Buddy Allocator example

- Want to allocate 7000 bytes
- Gets rounded up to power of two, 8192 (8k)
- Look for free 8k block. If found just hand it out. If not, bump to 16k try again
- No 16k free, bump to 32k
- 32k found! Break it up to four 8k chunks, hand out one
- Later if that 8k chunk is freed, if nearby chunks are also free, merge them together to create larger chunk
- This is designed to limit fragmentation



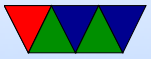
Linux – SLAB/SLOB/SLUB

- Have cache of commonly allocated structs
- Don't completely clear/free them when done, but leave them pre-initialized



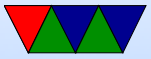
Linux – dynamic allocation (stack)

- Most code will try to do things on stack if possible
- Kernel stacks are small (why? Need to be contiguous, fragmentation, etc)
- Back in the day they tried to fit in one 4k page, not always work 8k. 16k now?
- Part of the problem was large structs being allocated, but especially deep callchains. Sometimes be fine in common case but then some obscure thing calls filesystem/network/network-card/etc and it all adds up



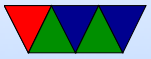
Linux – dynamic allocation (other)

- kmalloc()
- get_free_pages()
- vmalloc() – allocated virtual memory, avoids fragmentation



Linux – Memory Zones

- Not all memory is equal
- Not all is in reach of DMA
- 32-bit processes can't access memory above 4G
- Normal vs HIMEM (historical on Linux?)
- NUMA – sometimes want allocations to be close to CPU core



Linux – Memory stats

- /proc/buddyinfo
- /proc/zoneinfo