CS 5204
Operating Systems
Lecture 10

Godmar Back

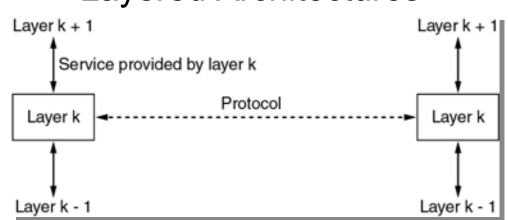
Announcements

- This week: first milestone meeting
  - Sign up now
  - Reports due by 5pm today
- Midterm will be handed back next Monday (Oct 31)
- Might switch some presentations - will update reading list and send email to affected presenters
  - Move up SFI (John) & Nooks (Hari)

Plan for Today

- End-to-End argument
- Review memory management

Layered Architectures



(horizontal component)

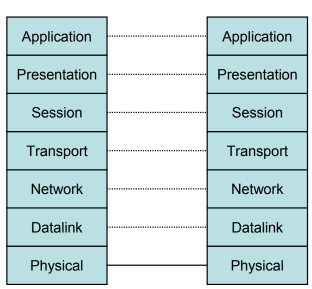
- Layer *k* may interact with peer layer *k* only via protocols

(vertical component)

- Layer *k+1* interacts with layer *k* via interface

Layering and the E2E Argument

- In any system using layering, designer has a choice of where to place functionality
  - Unless design by committee



End-to-end argument

- If correct & complete implementation requires help & knowledge only endpoints have, do not push the functionality down into lower layers
- Corollary:
  - A layer should only implement functionality that is needed by all clients, and can be completely implemented within that layer.

E2E Examples

- Careful file transfer
- Security & Encryption
- Error detection & correction
- Causal message delivery

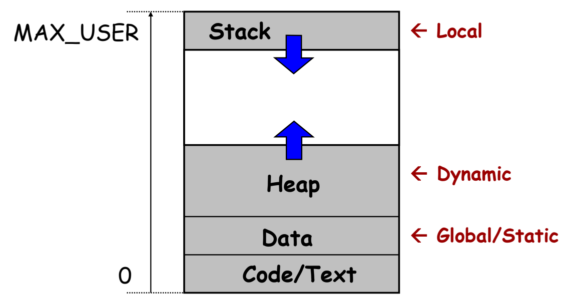
E2E (cont'd)

- Note that endpoint != application
  - Endpoint can also be a layer
  - How to identify the endpoints?
- Reasons for violating E2E:
  - Performance
  - Cost
  - Software engineering/Code Reuse (?)
- E2E is only a guiding principle, a type of "Occam's Razor"

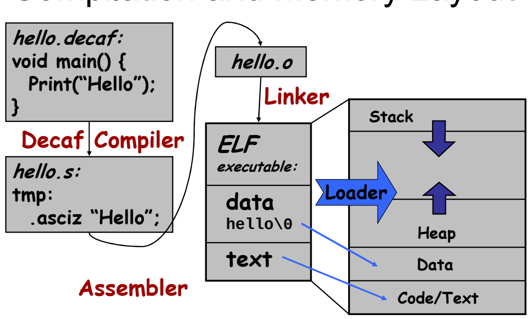
Review Memory Management

- Logical Organization
- Physical Organization
- Protection
- Paging

Logical Organization

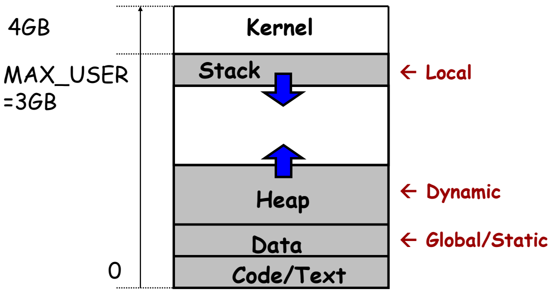


Compilation and Memory Layout

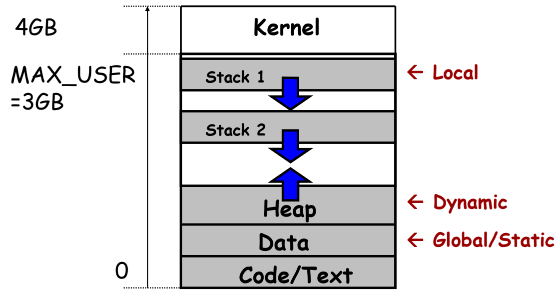


- Quick Demo of /proc/*/maps

Linux IA32 3+1 Model



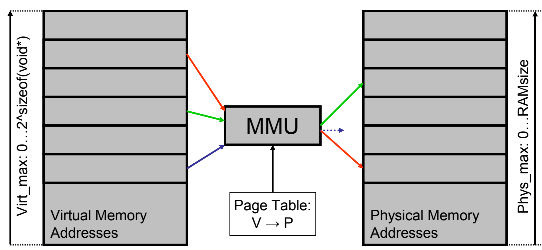
Linux IA32 3+1 Model (MT)



Virtual Address Space Management

- Key resource is virtual addresses
  - Memory objects occupy continuous regions:
    - Stacks of kernel threads
    - Data segment
    - Shared libraries:
      - Data, bss, text
    - Subtract space taken up by kernel if necessary
- 64bit architectures where sizeof (void*) == 8 this is practically no longer an issue
  - But tough on 32bit architectures

Virtual to Physical Mapping



OS Tasks for Virtual Memory Mgnt

- Predominant technology today: Paging
  - Older technique: segmentation
- Must maintain & update permissions for each page (Protection)
- Must maintain & update Virt→Phys mappings
  - and adjust on context switch
- Must manage physical frames
  - Detect usage & store content of unused frames to secondary storage

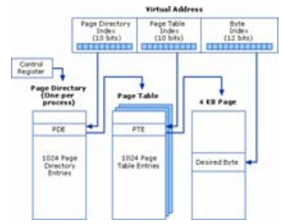
Protection

- Memory is associated with protection bits:
  - R – Read
  - W – Write
  - X – Execute
  - U – User – can it be accessed in user mode
- Oddball X86: pages only have “W” write bit + U bit. Recent addition “NX” bit – “don’t execute”
- Note: OS has full control over what a virtual address resolves to for a given process
  - Trap is caused if it doesn’t resolve to anything at the moment
  - Trap is caused if permissions don’t allow attempted action

Maintaining V → P Mappings

- Note that V → P translation happens on every memory access
  - Caching needed → cache is called TLB ("translation lookaside buffer")
  - What happens on a TLB miss?
- TLB misses can be handled in software or hardware:
  - Software: special handler must look up mapping and restore it (MIPS, Alpha)
  - Hardware: (x86) must follow data structure layout in memory so MMU can read information from there to refill TLB

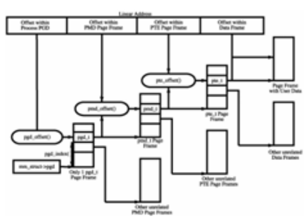
x86 Virtual Address Translation



- Pre-PAE x86

Microsoft Technet [URL]

Linux Page Table Structure



PAE x86 shown: from two-level to three-level translation scheme (2+9+9+12 vs 10+10+12)

Source:
Mel Gorman: Understanding the Linux Virtual Memory Manager

- Linux's internal structures mimic what MMU expects
  - Ports to other architectures must emulate x86 page tables in Linux kernel

TLB & Context Switches

- Note that mappings are process-specific:
  - Virtual address 10000 in process A maps to a different page then address 10000 in process B – every process gets it own address space
  - Either tag TLB entries with process id (MIPS) or flush entire TLB on context switch (x86)
    - The subsequent TLB misses are what makes a context switch expensive.

Managing Physical Frames

- Now memory is viewed as a resource
  - Must keep track of what's free, what's in use
    - Buddy allocator
  - Must keep track of what's stored in pages that are in use
- Consumers are memory objects
  - User process objects (stacks, data, text, …)
  - In-kernel ones (file system buffer caches, packet buffers, etc.)
- Algorithms
  - Idea: monitor how pages are accessed, write least used ones to disk
  - LRU heuristics, page buffering

Summary

- Memory Management involves
  - Logical organization of address space
  - Physical organization of memory
  - Policies for protection (page-based, traditionally)
  - Policies for paging