CSE 4733/6733 - Operating System 1

S. Torri, T. Ritter

Mississippi State University

CSE 4733/6733 - Operating System 1

S. Torri, T. Ritter

Mississippi State University

Linear Page Tables

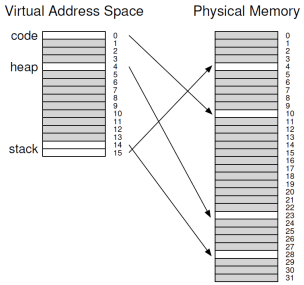


Figure 20.1: **A 16KB Address Space With 1KB Pages**

- Typically have one page table for every process, IE. 100 processes means 100 page tables
- Simple linear table is too memory intensive

Linear Page Tables
Typically have one page table for every process, IE. 100 processes means 100 page tables
Simple linear table is too memory intensive

1. Can we just make the page bigger, to make the table smaller?
   - Waste within pages - IE. Internal fragmentation
   - End up with MANY fewer jobs in-memory and memory quickly fills up
   - This is a tuneable parameter for a kernel
   - Not that simple...

Linear Page Tables

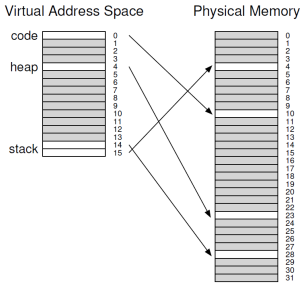
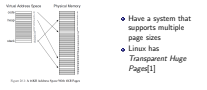


Figure 20.1: **A 16KB Address Space With 1KB Pages**

- Have a system that supports multiple page sizes
- Linux has *Transparent Huge Pages*[1]

Linear Page Tables



1. Complications?
   - A single TLB entry will map to a much larger amount of virtual memory in turn reducing the TLB misses.
   - Now use a *khugepaged* daemon to scan memory and collapse sequences of std pages into huge pages
   - Might be worth the effort for a database or other special purpose system, that why we have the libraries in Linux[2]
   - Virtual memory management becomes more complicated, latency spikes as OS moves data around and updates page entries...

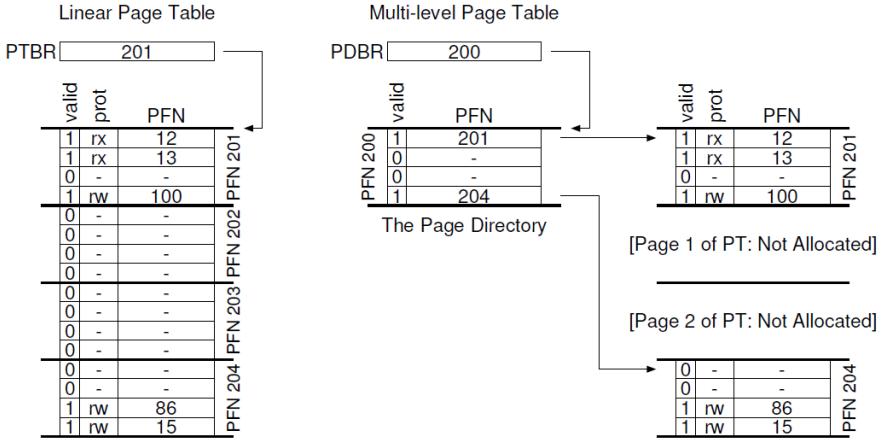
Hybrid Page Table

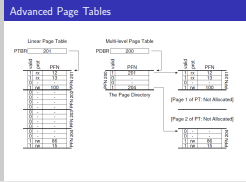
- Create a hybrid page table with both segmentation and paging
- User address space broken into a number of segments, with each segment into pages
- Both have strengths, paging to remove external fragmentation, segmentation lends itself to protection and sharing

Hybrid Page Table
- Create a hybrid page table with both segmentation and paging
- User address space broken into a number of segments, with each segment into pages
- Both have strengths, paging to remove external fragmentation, segmentation lends itself to protection and sharing

1. Comments
   - Easy base address and length calculation to protect memory bounds
   - Combined segmentation and paging has advantages, but brings back external fragmentation

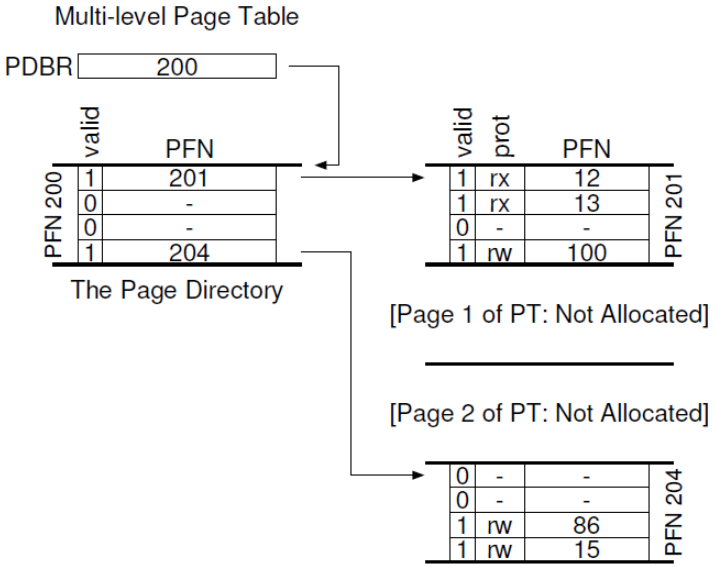
Advanced Page Tables

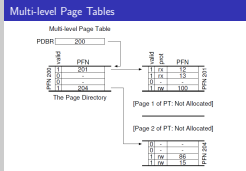




1. Note: Divide the page table into page sized chunks
   - We now have a valid flag, and a protection flag
   - If an entire page of page-table entries (PTEs) is invalid, save memory and don't allocate that page of the page table.
   - The page directory data structure results in a multi-level design with a much smaller memory footprint
   - Page directory contains either where the page table is, or if an entire page of the page table has no valid pages

Multi-level Page Tables





1. Note: Classic Trade-Off
   - We created a OS data structure with a smaller memory footprint
   - We see that a TLB *MISS* is more expensive than before
   - More page table levels are possible, what if the Page Directory is too large for a single page???
   - Supports sparse address spaces well, and page-table space is proportional to address space
   - Subtle note, valid flag in a Page Pirectory mean there is at least one page entry valid in the page where it points, unlike in the page table itself
   - More complexity in the design...

Inverted Page Table

- Create a *single* OS page table that has one entry for each real memory frame (physical page)
- Entry contains PID and what virtual page maps to that frame
- Create a simple hash function to speed up the linear search
- Inverted since the table in indexed by real memory frames, not virtual
- Page tables are a data structure that allows for trade-offs

Inverted Page Table
- Create a single OS page table that has one entry for each real memory frame (physical page)
- Entry contains PID and what virtual page maps to that frame
- Create a simple hash function to speed up the linear search
- Inverted since the table in indexed by real memory frames, not virtual
- Page tables are a data structure that allows for trade-offs

1. Requirements
   - Table requires PID of the process with the frame
   - Table requires control bits for valid, modified (dirty), protection, etc.
   - Table requires a chain pointer since your hash may point to the same entry as another virtual address

Swapping

- Multiprogramming, large jobs all drive an OS to support swapping
- Swap space is secondary storage, typically a disk address/chunk of space
- Need another flag - a *present bit* to determine if a page is in memory or swap
- The terminology of swapping - A *page fault* is when the OS wants to accesse a page that is not in memory

Swapping

- Multiprogramming, large jobs all drive an OS to support swapping
- Swap space is secondary storage, typically a disk address/chunk of space
- Need another flag - a *present bit* to determine if a page is in memory or swap
- The terminology of swapping - A *page fault* is when the OS wants to accesse a page that is not in memory

1. Comments
   - Page faults are expected, really more of a page miss
   - Hardware will raise an exception, and the OS will have to swap the page requested into memory
   - The OS will have a page-fault handler to service page faults

Swapping

- Swapping is by nature a *slow* process compared to memory access times, the TLB is a memory cache
- This is not a hardware function, rather the OS is going to do it
- When memory is full, you have to have a Page Replacement Policy

Swapping
- Swapping is by nature a slow process compared to memory access times, the TLB is a memory cache
- This is not a hardware function, rather the OS is going to do it
- When memory is full, you have to have a Page Replacement Policy

1. Comments
   - Page faults are expected, really more of a page miss
   - Hardware will raise an exception, and the OS will have to swap the page request into memory
   - The OS will have a page-fault handler to service the page fault

Swapping

- Kernel parameters of: *high watermark (HW) and low watermark (LW)*
- Linux has a *kswapd* to swap pages as needed
- When low watermark is reached then kswapd wakes to reclaim memory in the background.
- It stays awake until free memory reaches the high watermark.

Swapping

- Kernel parameters of: *high watermark (HW) and low watermark (LW)*
- Linux has a *kswapd* to swap pages as needed
- When low watermark is reached then kswapd wakes to reclaim memory in the background.
- It stays awake until free memory reaches the high watermark.

1. Comments
   - Linux kernel tuning is a complicated enterprise [3] eg. sysctl -a
   - The article quoted 49 Virtual Memory parameters out of 1,200 you can tune
   - Linux has a 3-level page table memory managment scheme and is complex
   - Swapping is normal, but low memory can make a machine work at the speed of disk I/O rather than memory I/O

References I


Free Software Foundation.
Transparent hugepage support, 2023.


Erik Rigtorp.
Using huge pages on linux, 2020.


Al Williams.
Fine tuning linux virtual memory, 2022.

References I

Free Software Foundation.
Transparent hugepage support, 2023.
Erik Rigtorp.
Using huge pages on linux, 2020.
Al Williams.
Fine tuning linux virtual memory, 2022.