

CS 370: OPERATING SYSTEMS
[MEMORY MANAGEMENT]

Shrideep Pallickara
Computer Science
Colorado State University

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.1

Frequently asked questions from the previous class survey

- Virtual addresses

October 25, 2018 CS370: Operating Systems [Fall 2018]
Professor: SHRIDEEP PALICKARA Dept. Of Computer Science, Colorado State University L20.2

Topics covered in this lecture

- Contiguous memory allocations
- Fragmentations
 - External and Internal
- Segmentation
- Paging

October 25, 2018 CS370: Operating Systems [Fall 2018]
Professor: SHRIDEEP PALICKARA Dept. Of Computer Science, Colorado State University L20.3

Each process is contained in a single continuous section of memory

CONTIGUOUS MEMORY ALLOCATION

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.4

Partitioning of memory

- Main memory needs to **accommodate** the OS and user processes
- Divided into two partitions
 - Resident OS
 - Usually low memory
 - User processes

October 25, 2018 CS370: Operating Systems [Fall 2018]
Professor: SHRIDEEP PALICKARA Dept. Of Computer Science, Colorado State University L20.5

Memory Mapping and Protection

- Base register (also referred to as a relocation register)
 - Smallest physical address
- Limit register
 - Range of logical addresses

October 25, 2018 CS370: Operating Systems [Fall 2018]
Professor: SHRIDEEP PALICKARA Dept. Of Computer Science, Colorado State University L20.6

Memory Mapping and Protection

- When CPU scheduler selects a process for execution
 - Base and limit registers reloaded as part of context switch
- Every address generated by the CPU
 - Checked against the relocation/limit registers

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.7

Memory Mapping and Protection

```

    graph LR
        CPU[CPU] -- Logical address --> Comp{<}
        LR[limit register] --> Comp
        Comp -- YES --> Add{+}
        BR[base register] --> Add
        Add -- Physical address --> Mem[memory]
        Comp -- NO --> Trap[TRAP to OS: Addressing ERROR]
    
```

E.g: base/relocation=100040 and limit=74600

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.8

Memory Allocation: Fixed Partition method

- Divide memory into several **fixed-size** partitions
 - Each partition contains exactly one process
- Degree of multiprogramming
 - Bound by the number of partitions

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.9

Memory allocation: Variable-partition method [1/2]

- Used in batch environments
- OS maintains table tracking memory utilization
 - What is available?
 - Which ones are occupied?
- Initially all memory is available
 - Considered a large **memory gap**
 - Eventually **many** memory gaps will exist

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.10

Memory allocation: Variable-partition method [2/2]

- OS orders processes according to the scheduling algorithm
- Memory allocated to processes until requirements of the next process cannot be met
 - Wait till a larger block is available
 - Check if smaller requirements of other processes can be met

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.11

Variable-partition method: Reclaiming spaces

- When process arrives if space is too large
 - Split into two
- When process terminates
 - If released memory is adjacent to other **memory gaps**
 - Fuse to form a larger space

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.12

Splitting and Fusing Memory spaces

October 25, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.13

Dynamic Storage Allocation Problem

- Satisfying a request of size n from the set of available spaces
 - First fit
 - Best fit
 - Worst fit

October 25, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.14

First fit

- Scan list of segments until you find a memory-gap that is big enough
- Gap is broken up into two pieces
 - One for the process
 - The other is unused memory

October 25, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.15

Best Fit

- Scan the entire list from beginning to the end
- Pick the smallest memory-gap that is adequate to host the process

October 25, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.16

Comparing Best Fit and First Fit

- Best fit is **lower** than first fit
- Surprisingly, it also results in more **wasted memory** than first fit
 - Tends to fill up memory with tiny, useless gaps

October 25, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.17

Worst fit

- How about going to the other extreme?
 - Always take the largest available memory-gap
 - Perhaps, the new memory-gap would be useful
- Simulations have shown that worst fit is not a good idea either

October 25, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.18

SEGMENTATION

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.19

Base and limits translation lacks many of the features needed to support modern programs

- Base and limits translation supports only **coarse-grained** protection at the level of the *entire* process
 - It is not possible to prevent a program from **overwriting** its own code, for example
 - It is also **difficult to share** regions of memory between two processes
 - Since the memory for a process needs to be contiguous ...
 - Supporting dynamic memory regions, such as for heaps, thread stacks, or memory mapped files, becomes difficult to impossible

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.20

In our discussions so far ...

- Logical/virtual memory is **one-dimensional**
 - Logical addresses go from 0 to some max value
- Many problems can benefit from having two or more **separate** logical address spaces

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.21

A compiler has many tables that are built up as compilation proceeds

- Source Text
- Symbol table
 - Names and attributes of variables
- Constants Table
 - Integer and floating point constants
- Parse tree
 - Syntactic analysis of program
- Stack
 - Procedure calls within the compiler

} Grows continuously as compilation proceeds

} Grows and shrinks in unpredictable ways during compilation

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.22

One dimensional address space with growing tables

Program has an exceptional number of variables

Address space allocated to the constant table

Address space being used

Free

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.23

One dimensional address space with growing tables

Program has an exceptional number of variables

Address space allocated to the constant table

Address space being used

Free

Symbol table has BUMPED INTO the source text table

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.24

Options available to the compiler

- Say that compilation cannot continue
 - Not cool
- Play Robin Hood
 - Take space from tables with room
 - Give to tables with little room

October 25, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.25

What would be really cool ...

- Free programmer from having to manage expansion and contraction of tables

October 25, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.26

But how?

- Provide **many** completely **independent address spaces**
 - Segments
- Each segment has linear sequence of addresses
 - 0 to max

October 25, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.27

Segments and Base/Limit registers

- The hardware supports an **array** of pairs of base and bounds registers, for each process
 - **Segment Table**
- Each entry in the array controls a portion, or **segment**, of the virtual address space
- The physical memory for each segment is stored contiguously, but different segments can be stored at different locations
 - For example, code and data segments are not immediately adjacent to each other in either the virtual or physical address space

October 25, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.28

Other things about segments

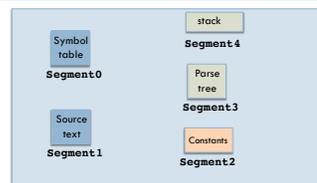
- Different segments can and do have different lengths
- Segments grow and shrink independently without affecting each other. For example, consider a segment for the stack
 - Size increase: something pushed on stack segment
 - Size decrease: something popped off of stack segment

October 25, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.29

Users view memory as a collection of variable-sized segments



October 25, 2018
Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L20.30

Segmentation

- Logical address space is a collection of segments
- Segments have name and length
- Addresses specify
 - Segment name
 - Offset within the segment
- Tuple: $\langle \text{segment-number}, \text{offset} \rangle$

October 25, 2018
 Professor: SHRIDEEP PALICKARA
 CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University
 L20.31

Segmentation Addressing Example

Segment	Limit	Base
0	1000	1400
1	400	6300
2	400	4300
3	1000	3200
4	1000	4800

October 25, 2018
 Professor: SHRIDEEP PALICKARA
 CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University
 L20.32

Segmentation Hardware

October 25, 2018
 Professor: SHRIDEEP PALICKARA
 CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University
 L20.33

FRAGMENTATION

October 25, 2018
 CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University
 L20.34

Contiguous Memory Allocation: Fragmentation

- As processes (and segments) are loaded/removed from memory
 - Free memory space is **broken** into small pieces
- External fragmentation**
 - Enough space to satisfy request; BUT
 - Available spaces are **not contiguous**

October 25, 2018
 Professor: SHRIDEEP PALICKARA
 CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University
 L20.35

Fragmentation: Example

October 25, 2018
 Professor: SHRIDEEP PALICKARA
 CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University
 L20.36

Fragmentation can be internal as well

- Memory allocated to process may be *slightly larger* than requested
- Internal fragmentation**
 - Unused memory is internal to blocks

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.37

Compaction: Solution to external fragmentation

- Shuffle** memory contents
 - Objective: Place free memory into large block
- Not possible if relocation is static
 - Load time
- Approach involves moving:
 - Processes towards one end
 - Gaps towards the other end

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.38

Compaction: Example

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.39

Memory compaction is time intensive and is usually not done

- Let's consider a machine with 1 GB of RAM
- The machine can copy 4 bytes in 20 nanoseconds
- Time to compact all the memory?
 $10^9 \times (20 \times 10^{-9} / 4) = 5$ seconds (approximately)
 Note: 1 GB is approximately 10^9 bytes.

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.40

PAGING: OVERVIEW OF THE MAPPING PROCESS

October 25, 2018

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.41

Overview of how mapping of logical and physical addresses is performed

October 25, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L20.42

Noncontiguous memory management

PAGING

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.43

The Paging memory management scheme

- Physical address space of process can be **non-contiguous**
- Solves problem of fitting variable-sized memory chunks to backing store
 - Backing store has fragmentation problem
 - Compaction is impossible

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.44

Basic method for implementing paging

- Break memory into **fixed-sized blocks**
 - Physical memory: **frames**
 - Logical memory: **pages**

} Same size
- Backing store is also divided the same way

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.45

What will seem odd, and perhaps cool, about paging [1/2]

- While a program **thinks of its memory as linear ...**
 - It is usually **scattered** throughout physical memory in a kind of abstract mosaic
- The processor will execute one instruction after another using virtual addresses
 - The virtual addresses are still linear
 - However, an instruction located at the end of a page will be located in a **completely different region** of physical memory from the next instruction at start of another page

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.46

What will seem odd, and perhaps cool, about paging [2/2]

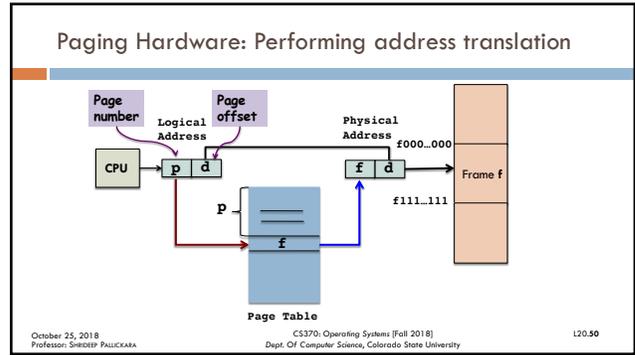
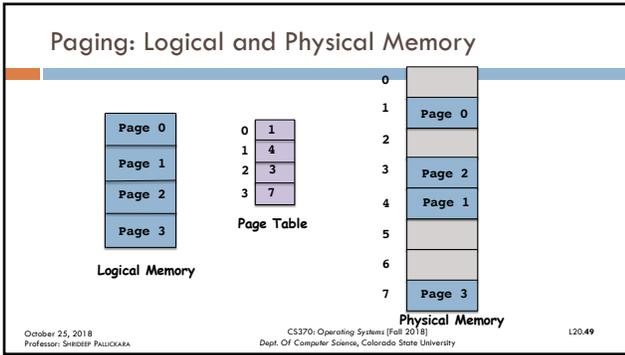
- Data structures appear to be contiguous using virtual addresses
 - But a large matrix is scattered across many physical page frames

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.47

Paging: Analogy

- Shuffling several decks of cards together
- A single process in its virtual address page sees the cards of a single deck in order
 - A different process sees a completely different deck, but it will also be in order
- In physical memory, however, the **decks of all processes** currently running will be **shuffled** together, apparently at random
- Page tables are the magician's assistant in locating cards from the shuffled decks

October 25, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L20.48



The contents of this slide-set are based on the following references

- Avi Silberschatz, Peter Galvin, Greg Gagne. *Operating Systems Concepts, 9th edition*. John Wiley & Sons, Inc. ISBN-13: 978-1118063330. [Chapter 8]
- Andrew S Tanenbaum and Herbert Bos. *Modern Operating Systems, 4th Edition*, 2014. Prentice Hall. ISBN: 013359162X/ 978-0133591620. [Chapter 3]
- Thomas Anderson and Michael Dahlin. *Operating Systems Principles and Practice, 2nd Edition*. Recursive Books. ISBN: 978-0985673529. [Chapter 8]

October 25, 2018
 Professor: SHRIDEEP PALICKARA
 CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University
 L20.51