A Non-Exhaustive Exam 2 Study Guide*

October 25, 2024

Chapter 9: Deadlocks

A deadlock occurs when a set of processes are blocked, each waiting for a resource held by another process in the set.

Resources

1. Preemptable: one that can be taken away from the process owning it with no ill effects, e.g. Memory
2. Nonpreemptable: one that cannot be taken away from its current owner without potentially causing failure.

Deadlocks involve nonpreemptible resources.

Conditions for Deadlock

Four conditions must be present for a deadlock to occur:

1. Mutual exclusion: Only one process can hold a resource at a time.
2. Hold-and-wait: Processes holding resources can request new resources.
3. No-preemption: Resources cannot be forcibly taken away.
4. Circular wait: A circular chain of processes waiting for resources.

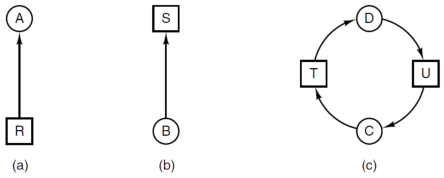
Deadlock Modeling

Deadlocks can be modeled using directed graphs:

- Circles: Processes.
- Squares: Resources.

*May not cover ever single thing on the exam. Use this as one of the tools in your study strategy, not the only tool.

- Arcs: Indicate resource requests and allocations.



(a) Holding a resource. (b) Requesting a resource. (c) Deadlock.

Arrow from process to resource: process holds the resource
Arrow from process to resource: process wants the resource

Deadlock Strategies

- Ostrich algorithm: Ignore the problem.
- Detection and recovery: Let deadlocks occur, detect them, and recover.
- Dynamic avoidance: Avoid deadlocks by careful resource allocation.
- Prevention: Prevent one of the four conditions from occurring.

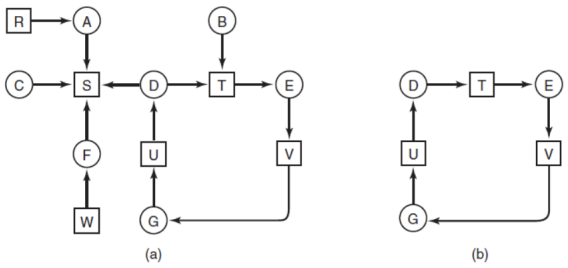
Deadlock Detection and Recovery

Know how to work both of these:

- Single resource of each type: Use a resource graph and check for cycles.

  Consider a system with seven processes, A though G, and six resources, R through W. The state of which resources are currently owned and which ones are currently being requested is as follows:

  - Process A holds R and wants S.
  - Process B holds nothing but wants T.
  - Process C holds nothing but wants S.
  - Process D holds U and wants S and T.
  - Process E holds T and wants V.
  - Process F holds W and wants S.
  - Process G holds V and wants U.



The cycle, shown in b, shows the system is deadlocked.

- Multiple resources of each type: Use a matrix-based algorithm.

Recovery Techniques

- Preemption: Take away a resource from a process.
- Rollback: Restore a process to a previous state.
- Killing processes: Terminate a process to break the deadlock.

Deadlock Avoidance

Banker's algorithm: Checks if granting a resource request leads to an unsafe state. Difficult to implement in practice because processes rarely know their maximum resource needs in advance.

Deadlock Prevention

Deadlocks can be prevented by making it so one of the requirements for deadlocks is impossible.

- Attack mutual exclusion: Make resources sharable if possible.
- Attack hold-and-wait: Require processes to request all resources at once or release all held resources before requesting new ones.
- Attack no-preemption: Make resources preemptible if possible.
- Attack circular wait: Impose a total ordering on resources and require processes to request them in order.

Chapter 10: Memory Management

Modern operating systems use a memory hierarchy to manage memory efficiently. The hierarchy consists of:

- Cache memory: Very fast, expensive, and volatile.
- Main memory: Medium speed, price, and volatile.
- Disk storage: Slow, cheap, and nonvolatile.

The memory manager is responsible for:

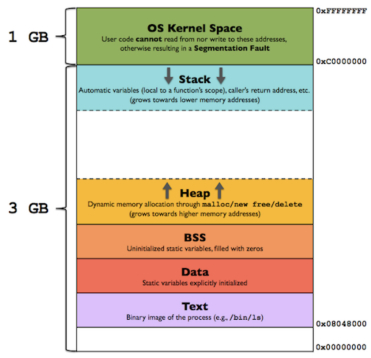
- Tracking memory usage.
- Allocating memory to processes.
- Deallocating memory when processes are done.

Early Systems

Early computer systems had no memory abstraction. This meant that: Every program saw physical memory. Two programs could not reside in memory at the same time due to address conflicts. To overcome this limitation, swapping was used. Swapping involved saving the entire contents of memory to disk and then loading a new program. However, swapping is inefficient and limits concurrency.

Address Spaces

To address the limitations of no memory abstraction, address spaces were introduced. An address space is an abstraction of memory that a process can use.



Each process has its own independent address space, which allows multiple programs to run concurrently without address conflicts. Threads share the address space of the process that created them.

Dynamic Relocation

Dynamic relocation is a technique that maps logical addresses to physical addresses at runtime. This is achieved using two special registers:

- Base register: Holds the starting physical address of the process's address space.
- Limit register: Holds the size of the process's address space.

Logical vs. Physical Addresses

- Logical addresses: The addresses generated by the CPU as the program is executed.
- Physical addresses: The addresses used to reference physical memory.

Dynamic relocation translates logical addresses to physical addresses.

Swapping

Swapping is a technique that brings entire processes into memory, runs them for a while, and then puts them back on disk. This allows processes larger than physical memory to run. However, swapping can be inefficient due to the overhead of transferring entire processes.

Heap Fragmentation

Heap fragmentation occurs when free memory space in the heap is divided into small, non-contiguous chunks. This can prevent the allocation of large memory blocks. There are two types of heap fragmentation:

- External fragmentation: When enough total free memory exists to satisfy a request, but it is not contiguous.
- Internal fragmentation: When more memory is allocated than is actually needed.

Compaction

Compaction is a technique that reorganizes fragmented memory space to make free space contiguous. However, compaction is an expensive operation.

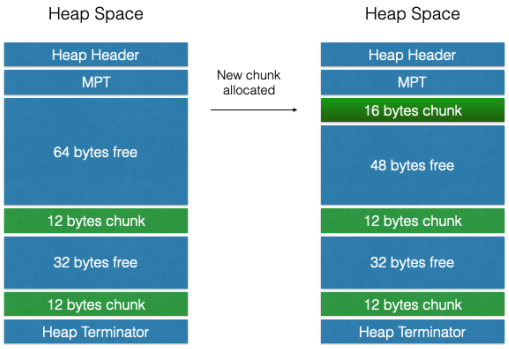
Allocation

The OS must decide how much memory to allocate to a process. If a process cannot grow, the OS can allocate exactly the amount needed. However, if a process can grow, the OS may allocate extra memory to reduce the overhead of moving or swapping processes.

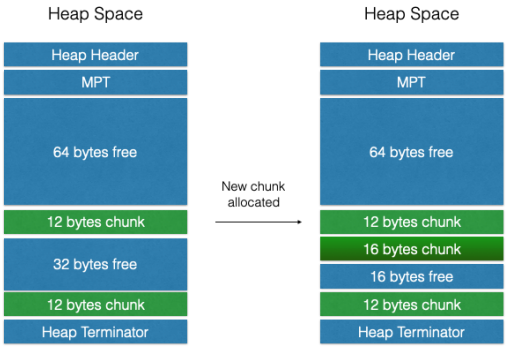
Allocation Strategies

There are several strategies for allocating memory:

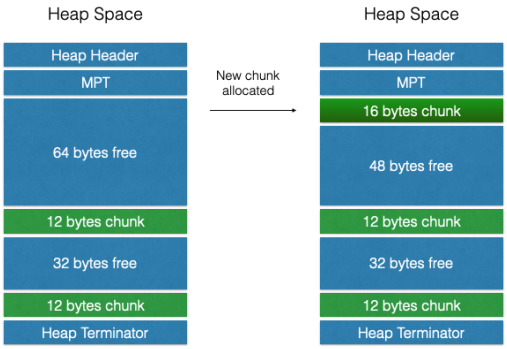
- First Fit: Allocate the first free block that is large enough.



- Next Fit: Similar to first fit, but starts searching from the location of the last allocation.
- Best Fit: Allocate the smallest free block that is large enough. Leaves the least amount of leftover space in the chosen block.



- Worst Fit: Allocate the largest free block.



Managing Free Memory

There are two main ways to track memory usage:

- Bitmaps: Each bit represents an allocation unit, with 0 indicating free and 1 indicating allocated. Most space efficient.
- Free lists: Linked lists of free memory segments. Quicker to find contiguous free blocks.

Overlays

Overlays are a technique used when program size exceeds physical RAM. The programmer specifies which parts of the program can be loaded and unloaded from memory as needed. However, overlays require special precautions and extra work by the programmer. Code in one overlay cannot reference code or data in another overlay since they aren't resident in memory at the same time.

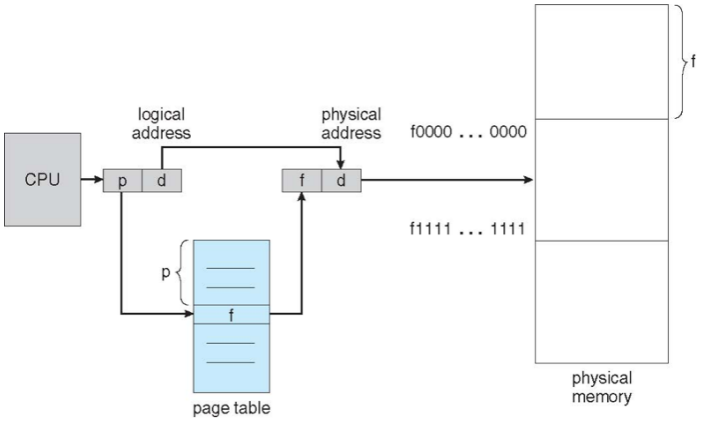
Paging

Paging is a memory management scheme that allows a process's logical address space to be non-contiguous.

- Frames: Fixed-sized blocks of physical memory.
- Pages: Fixed-sized blocks of logical memory.
- Page table: Maps logical addresses to physical addresses.

Page Size must equal Frame Size

Address Translation



Virtual address are split into two parts.

1. Upper: Index bits.
2. Lower: Offset bits.

For given logical address space 2^m and page size 2^n

page number	page offset
p	*d*
m - n	*n*

- m: The size of the virtual address space in bits / The CPU size in bits / The pointer size in bits
- n: The page size in bits / The frame size in bits
- m - n: The page table size in bits

Translation Lookaside Buffer (TLB)

A TLB is a cache that stores recent page table entries to speed up address translation. It searches all entries in parallel, making it very fast. However, TLBs are small due to their complex circuitry.

Effective Access Time

The effective access time (EAT) is the average time it takes to access a memory location.

With paging, the EAT depends on the hit ratio of the TLB.

- Parallel TLB: $EAT = (M + \epsilon)(\alpha) + (2M)(1 - \alpha)$
- Seriall TLB: $EAT = (M + \epsilon)(\alpha) + (\epsilon * 2M)(1 - \alpha)$

where:

- M is the memory cycle time.
- ϵ is the TLB lookup time.
- α is the TLB hit ratio.

Context Switches

When a context switch occurs, the TLB must be flushed because it typically does not store process-specific information. This can slow down the first few memory accesses of the new process. Some TLBs use address-space identifiers (ASIDs) to avoid flushing on context switches. [

Demand Paging

Demand paging is a technique that loads pages into memory only when they are needed. This allows processes to run even if they are not completely in memory.

- Page fault: Occurs when a process tries to access a page that is not in memory.
- Pager: The part of the OS that handles page faults.

Benefits of Demand Paging

- Programs can be larger than physical memory.
- More programs can run concurrently.
- Less I/O is needed.
- Programs can start faster.
- More users can be supported.

Page Table Entry

Each page table entry includes a valid-invalid bit, indicating whether the page is in memory.

Handling Page Faults

1. OS checks if the reference is valid.
2. If valid, OS finds an empty frame.
3. OS swaps the page into the frame.
4. OS updates the page table.
5. Process is restarted.

Multilevel Page Tables

Multilevel page tables are used to avoid keeping all page tables in memory at once. They divide the page table into multiple levels, with each level pointing to the next.

Working Set

The working set of a process is the set of pages it is currently using. The OS tries to keep the working set in memory to minimize page faults.

Page Replacement

When memory is full, the OS must choose a page to replace (evict) to make room for a new page.

Page Replacement Algorithms

- First-In, First-Out (FIFO): Replaces the oldest page. Simple but suffers from Belady's Anomaly.
- Optimal: Replaces the page that will not be used for the longest time. Impossible to implement in practice.
- Least Recently Used (LRU): Replaces the least recently used page. Difficult to implement efficiently.
- Not Recently Used (NRU): Approximates LRU using reference bits. Divides the pages into four classes and evicts a random page from the lowest class.
- Most Frequently Used (MFU): Evict the page with the highest reference count. It's probably been there the longest.
- Not Frequently Used (NFU): Evict the page with the lowest reference count. It's probably low because it's not used much.
- Second Chance: A variation of FIFO that gives pages a second chance if they have been referenced recently.

- Clock: A circular buffer implementation of Second Chance.

Allocation of Frames

The OS must allocate frames to processes. Two major schemes are:

- Fixed allocation: Each process gets a fixed number of frames.
- Priority allocation: Frames are allocated based on process priority.

Thrashing

Thrashing occurs when a process spends more time swapping pages than executing instructions. This can happen if the process does not have enough frames.

Chapter 12: File Systems

A file system provides a way to organize and store data on secondary storage devices like disks. It allows users to:

- Create, delete, and modify files.
- Organize files in directories.
- Share files with other users.

Requirements File Systems

1. It must be possible to store a very large amount of data.
2. The information must survive the termination of the process that creates it.
3. Multiple processes must be able to access the information at once.

File Concept

A file is a contiguous logical address space that can store various types of data. Files can be:

- Data files: Contain numeric, character, or binary data.
- Program files: Executable code.

File Structure

Files can have different structures, including:

- No structure: A sequence of words or bytes.
- Simple record structure: Fixed-length or variable-length records.
- Complex structures: Formatted documents, relocatable load files.

File Operations

Common file operations include:

- Create: Create a new file.
- Write: Write data to a file.
- Read: Read data from a file.
- Seek: Reposition within a file.
- Delete: Remove a file.
- Truncate: Reduce the file size to zero.
- Open: Prepare a file for access.
- Close: Finish accessing a file.

Access Methods

- Sequential access: Data is accessed in a linear order.
- Direct access: Data can be accessed directly by block number.

Directory Structure

A directory is a collection of nodes containing information about files. Directories provide a way to organize files into a hierarchical structure. [

Directory Organizations

Single-level directory: All files are in one directory. Suffers from naming and grouping problems. Two-level directory: Each user has their own directory. Solves the naming problem but not the grouping problem. Tree-structured directories: Files are organized in a hierarchical tree structure. Supports efficient searching and grouping. Directed acyclic-graph directories: Allows shared subdirectories and files. Can lead to dangling pointers if not handled carefully.

File-System Structure

A file system is typically organized into layers:

- Application programs: Interact with the file system through system calls.
- Logical file system: Provides a high-level view of the file system.
- File-organization module: Manages the physical organization of files on disk.
- Basic file system: Interfaces with the device driver.
- Device driver: Controls the physical device.

File-System Implementation

- Boot control block: Information needed to boot the OS.
- Volume control block: Details about the volume (partition).
- File control block (FCB): Contains information about each file.

Virtual File Systems (VFS)

VFS provides an object-oriented way to implement file systems. It allows the same system call interface to be used for different types of file systems.

Directory Implementation

- Linear list: Simple but slow for searching.
- Hash table: Faster searching but can suffer from collisions.

Allocation Methods

- Contiguous allocation: Each file occupies a set of contiguous blocks. Simple but can lead to fragmentation and external fragmentation.
- Linked allocation: Each file is a linked list of blocks. No external fragmentation but slow for random access.
- Indexed allocation: All pointers to a file's blocks are stored in an index block. Supports random access but has overhead for the index block.

Free-Space Management

- Bit vector: Each bit represents a block, with 0 indicating free and 1 indicating allocated.
- Linked list: A list of free blocks.

Recovery

3-2-1 Rule

- 3 copies of every files
- 2 different storage mediums
- 1 offsite

Consistency checking: Compares directory structure with data blocks and fixes inconsistencies.

Backups: Copying data to another storage device for disaster recovery.

Log Structured File Systems

Journaling file systems record file system updates as transactions in a log. Transactions are committed once written to the log, ensuring consistency even in case of a crash.

Example File Systems

MS-DOS File System (FAT)

First used in IBM PCs.

Uses a File Allocation Table (FAT) to track disk blocks.

Limited file size and partition size.

Three versions: FAT-12, FAT-16, FAT-32.

Ext2 File System

Used in Linux systems.

Uses inodes to store file information.

Supports large files and partitions.

Structure of Ext2

- Boot block: Reserved for boot loaders.
- Superblock: Contains global file system information.
- Group descriptors: Describe each block group.
- Block and inode bitmaps: Track free blocks and inodes.
- Inode table: Stores inode structures.
- Data blocks: Hold file data.

CD-ROM File System (ISO 9660)

Designed for read-only media.

No provision for free space management.

Standardized format for CD-ROMs.

Chapter 13: File-System Management and Optimization

Disk-Space Management

Strategies for storing files: Allocate consecutive blocks. [86] Split files into non-contiguous blocks. [86]

Block Size

Choosing the right block size is important for performance and space efficiency. [86] Large blocks reduce seek time but waste space for small files. [86] Small blocks reduce wasted space but increase seek time. [87]

Managing Free Blocks

Linked list: Uses a block to store pointers to free blocks. [87] Bitmap: Uses a bit for each block, with 0 indicating free. [88]

Disk Quotas

Disk quotas limit the amount of disk space a user can use.

File System Backups

Backups are essential for disaster recovery and protecting against data loss.

Types of Backups

Full backup: Copies the entire file system. Incremental backup: Copies only files modified since the last backup.

Backup Strategies

Physical dump: Copies blocks sequentially.

Logical dump: Copies files and directories based on modification time.

File-System Consistency

Consistency checking ensures that the file system is in a valid state. It involves checking for:

- Missing blocks.
- Duplicate blocks.
- Inconsistent link counts.