

Memory Management

Robert Dickerson

- Why is it generally inefficient to allow only one process to be in memory at a time?

Answer

- If the single process blocks for I/O, no other processes can use the processor

- What would happen if a system allowed many processes to be placed in main memory, but did not divide memory into partitions?

Answer

- The processes would all share their memory
- Any malfunctioning or malicious process could damage any or all of the other processes

- Name the two types of **fetch strategies**
- Describe when one might be more appropriate than the other

- **Demand fetch and anticipatory fetch**
- If the system cannot predict future memory usage with accuracy, then the lower overhead of demand fetching results in higher performance and utilization
- If program exhibits predictable behavior, anticipatory fetch strategy is best

Contiguous vs. Noncontiguous Memory

- When is **noncontiguous** preferable to **contiguous** memory allocation?

Answer

- **Noncontiguous** is better when available memory contains no area large enough to hold the incoming program in a contiguous piece

- What sort of overhead is involved in a **noncontiguous memory allocation scheme?**

- You need to store:
 - available blocks
 - blocks that belong to separate processes
 - where those blocks reside in memory

- Describe the benefits and drawbacks of large and small partition sizes

- **Larger partitions** allow large programs to run, but result in internal fragmentation for small programs
- **Small partitions** reduce the amount of internal fragmentation and increase the level of multiprogramming by allowing more programs to reside in memory at once

- Explain the difference between **internal** and **external** fragmentation

- **Internal fragmentation** occurs in fixed-partition environments when a process is allocated more space than it needs.
- **External fragmentation** occurs in variable-partition environments when memory is wasted to holes developing in memory between partitions.

- Why are **segmentation/paging** systems appealing?

- Segmentation/paging offers architectural simplicity of paging and access control capabilities of segmentation

- Why are large page sizes more favorable in today's systems than they were decades ago?

- Cost of memory is cheaper, applications more memory intensive
- Cost of internal fragmentation is less of a concern
- Large pages require the system to perform fewer costly I/O operations

- What restriction does **assembly time** or **load time binding** of address place on the swapping of a process?

- If addresses are not bound at runtime, then the process must be swapped into the same memory location from which it was swapped out

- Describe 3 ways one can address the problem of **external fragmentation**

- **Compaction** - if addresses can be bound at runtime, we can move the used memory around so that the unused memory is contiguous and reset the base register.
- Break a process into 2 or more smaller sections to increase likelihood small memory spaces will be used
- Loosen the requirement that process memory be contiguous. Solution- paging

- What is a **TLB**?

- **Translation Look-aside Buffer**

- hardware method of speeding up translation of logical to physical memory addresses
- acts as a cache for more recent translations
- page number is compared to all page-frame pairs in parallel - very fast!

- What is involved with a **context switch**?

- Occurs when a process is interrupted and another process gains control of the system
- Save registers, program counter, page table or segment pointers, stack, etc.

- How does **segmentation** compare to **paging**?

- compiled into separate units (code, data, stack)
- user specifies a segment number and offset instead of an address.
- Segments can have variable lengths
 - requires a field in segment table containing the limits of the offset value

- What property must code have in able to be shared?

- It must be **reentrant**
- A computer program or routine is described as reentrant if it can be safely called recursively and concurrently from multiple processes.
- To be reentrant, a function must:
 - hold no static data
 - must not return a pointer to static data,
 - must work only on the data provided to it by the caller, and must not call non-reentrant functions.

- What is **virtual memory**?

- Allows execution of processes that may not be completely in memory.
- Allows programmers to think of memory as a large uniform space

- Why is virtual memory almost never used in a realtime system?

- Dependability of the system
- Predictable in time required to complete operations
- Virtual memory creates less predictable delays in memory-related operations

- What happens on a **page fault**?

- Page table for the process is used to determine if the page corresponding to logical address is invalid
- If it is, OS chooses a free frame or selects a page to be paged out, and gets the page from backing store
- Page table is updated to reflect that page is now in memory and instruction is restarted

- What is **Belady's Anomoly**?

- Belady's anomaly states it is possible to have more page faults when increasing the number of page frames while using FIFO method of frame management.
- Previously, it was believed that an increase in the number of page frames would always provide the same or fewer page faults

example

3 page frames

3 2 1 0 3 2 4 3 2 1 0 4

- Why is it that no page replacement algorithm can be optimal?

- Optimal algorithms would be where the page that will not be used for the longest time will be swapped out
- No algorithm can be optimal because it is impossible to foresee the future page usage

- How do you implement a **LRU page replacement?**

- Save with each page clock information. When a least recently page is needed, a comparison of all the timestamps are made
- Implement a stack containing references to all the pages. When a page is accesses, it is moved to the top of the stack. The page at the bottom of the stack is the LRU

- What is an **inverted page table**?

- Instead of mapping virtual page to a frame, map the frames to the pages
- Results in a table whose number of entries corresponds to the number of frames