CS460—Operating Systems

Lecture 5

Memory Management–recent systems (cont.)

Textbook: Operating Systems
by William Stallings
1. Demand Paging (continued)

• How do we swap a page out of memory?

• FIFO (First In First Out).
  FIFO removes the page that has been in the memory the longest.

• LRU (Least Recently Used).
  LRU removes the page that shows the least sign of recent usage.

• MRU (Most Recently Used).
  MRU removes the page that shows the strongest sign of recent usage.
• LFU (Least Frequently Used).
  LFU removes the page that shows the least amount of recent usage, over certain period of time.

• How do we make use of the PMT (Page Map Table)?

• How to improve the performance of demand paging?
  – **Working set**: a set of pages in memory which do not need to be swapped out back and forth.
  – However, identifying working set is not easy.

• Summary
  – 1. Virtual memory is introduced.
  – 2. Utilizes memory more efficiently.
  – 3. Overhead is heavy.
2. **Segmented Memory Allocation**

- Both of the paging algorithms divide a job into physically equal-sized pages, which might cause serious problems in reality.

- The idea of segmented memory allocation algorithm is to divide job into logical segments.

- Memory is consequently divided into page frames with different sizes → external fragmentation reappears.

- For each job we associate it with a Segment Map Table (SMT).
• Similar to paging we need to maintain the following data structures: Job Table, Segment Map Table and Memory Map Table.
  – 1. Job Table lists every job in process.
  – 2. Segment Map Table lists details about each segment.
  – 3. Memory Map Table monitors the allocation of main memory.

• How to access a specific instruction? You still need to locate SEGMENT NUMBER and DISPLACEMENT.
3. Segmented/Demand Paged Memory Allocation

• IDEA: Divide each segment further into pages of equal size. Hence we need the following 4 data structures:

  - 1. Job Table lists every job in process.
  - 2. Segment Map Table (for each job) lists details about each segment.
  - 3. Page Map Table (for each segment) monitors the pages associated with each segment.
  - 4. Memory Map Table monitors the allocation of main memory.
• Now we can move pages at will between main memory and second memory — **Virtual Memory**.

• **Advantage of Virtual Memory.**
  
  – 1. Job size has almost nothing to do with size of memory.
  
  – 2. Memory is used more efficiently.
  
  – 3. External fragmentation is eliminated and internal fragmentation is minimized.
  
  – 4. Sharing of code/data is possible.
  
  – 5. Dynamic linking of program segments is facilitated.

• **Disadvantage of Virtual Memory.**
  
  – 1. Hardware cost is increased.
  
  – 2. Overhead (for paging interrupts) is increased significantly.
  
  – 3. High cost for preventing thrashing.

• **End of Memory Management.** Before we say that, how does UNIX memory management work?
4. Why buddy system?

• In real systems, a page is usually of size $2^a$ — a power of 2.

• A combination of dynamic partition and paging.

• Allocation algorithm
  
  1. Take a free block $B_i$.
  2. If the job requests more than 50% of $B_i$, allocate $B_i$ to the job. Otherwise break $B_i$ into two blocks $B_{i1}, B_{i2}$ with equal size and proceed recursively on $B_{i1}$.

• Deallocation algorithm
  
  1. Take a block $B_i$ to be released.
  2. If the buddy of $B_i$ (i.e., shares the same parent and has the same size as $B_i$), $B_j$, is free, then combine $B_i, B_j$ into a free block $B_k$ with twice of the size.
  3. Recurse on $B_k$. 