Lecture 11

Memory Replacement Techniques

Preliminaries

- A process has a number of page frames allocated to it, called the *resident set*. If it is small, more processes can reside in memory at any one time. However, if the number of pages in memory is small, the page fault rate is high – beyond a certain size, additional memory will have no effect on the fault rate.
- When the main memory is fully allocated to user processes, a new request for a page frame is served by replacing one that is already allocated.
- One option is to replace only pages of the process itself, called *local replacement*, in contrast to selecting a page from all processes in the system, called *global replacement*.
- It is possible that a page frame is *locked*, meaning that the page stored in that frame can't be replaced – shared by several processes, I/O buffer, etc.

Page replacement policies

- *Belady's Min*: pick the page that will be used least soon. If t_i is the time at which page *i* will next be accessed, the selected page corresponds to $\text{argmax}_i t_i$. This strategy is optimal as it assures that the number of page faults is minimized. Of course, it is not a practical one (can't anticipate the future), but it can be used as a reference for simulations.

- *First In First Out* starts from the idea that pages are used for a finite amount of time after which they become "old". The page selected here is the one that has been in memory the longest. Implementation is done by a queue – all new pages are added to the tail of the queue.

- *Second chance* is an extension of FIFO: when a page is pulled off the head of the queue, the accessed (A) bit is examined. If it's 0, the page is swapped out, else the bit is cleared and the page is reinserted at the tail of the queue. A second examination of the queue will produce available pages.
- *The clock algorithm* is similar to the second chance: two clock hands are moving synchronously above pages; the distance between them determines how long the page is given to be accessed. If it has not been accessed within that time, it can be swapped out.
- In many implementations of the two-handed clock, the hands are not moved only when a page fault occurs. The pair of hands periodically advances some number of pages and keep a record of those with A = 0.

- *Not recently used* refers not only to pages with A=0 but also to pages which were not written into (M=0); therefore, if they have a copy on the disk, they don't need to be swapped out. Only one read brings the new page. This policy works well in conjunction with two-handed clock which analysis the bit pair AM – 00, and the lowest value is preferred.

- *Least recently used* is based on the time passed since a page was last time used. This concept is rarely used in practice because of the difficulty to implement it (software, hardware?). One possibility is to periodically mark pages as not present (P=0). When a page is accessed, it generates a page fault. The OS checks to see if the page is actually present, and if it is, the access time is recorded; P=1 to avoid further page faults.

- *Not frequently used* is based on counting memory references. Periodically, all pages in the memory are swept and for each one with A=1, A is cleared and a page counter is incremented. Then, the page with the smallest value of the counter is selected for swapping.
- This strategy penalizes newly loaded pages and keeps heavily used pages longer than wished.
- This policy can be improved by weighing recent references more heavily than older ones – this is called *aging*.

- Question: how can aging be implemented?

- Let’s assume a process has three page frames allocated. The process execution involves references to five pages, according to the following flow: 2,3,2,1,5,2,4,5,3,2,5,2

- Belady’s min 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2

2	2	2	2	2	2	4	4	4	2	2	2	page frame 1
	3	3	3	3	3	3	3	3	3	3	3	page frame 2
			1	5	5	5	5	5	5	5	5	page frame 3

- LRU

2	2	2	2	2	2	2	2	3	3	3	3
	3	3	3	5	5	5	5	5	5	5	5
			1	1	1	4	4	4	2	2	2

- FIFO

2	2	2	2	5	5	5	5	3	3	3	3
	3	3	3	3	2	2	2	2	2	5	5
			1	1	1	4	4	4	4	4	2

- *The working set* of pages corresponds to the number of pages a specific process is using at a time. There are variations, and two thresholds (called also *watermarks*) can be considered. They are the upper and lower bounds of the working set.

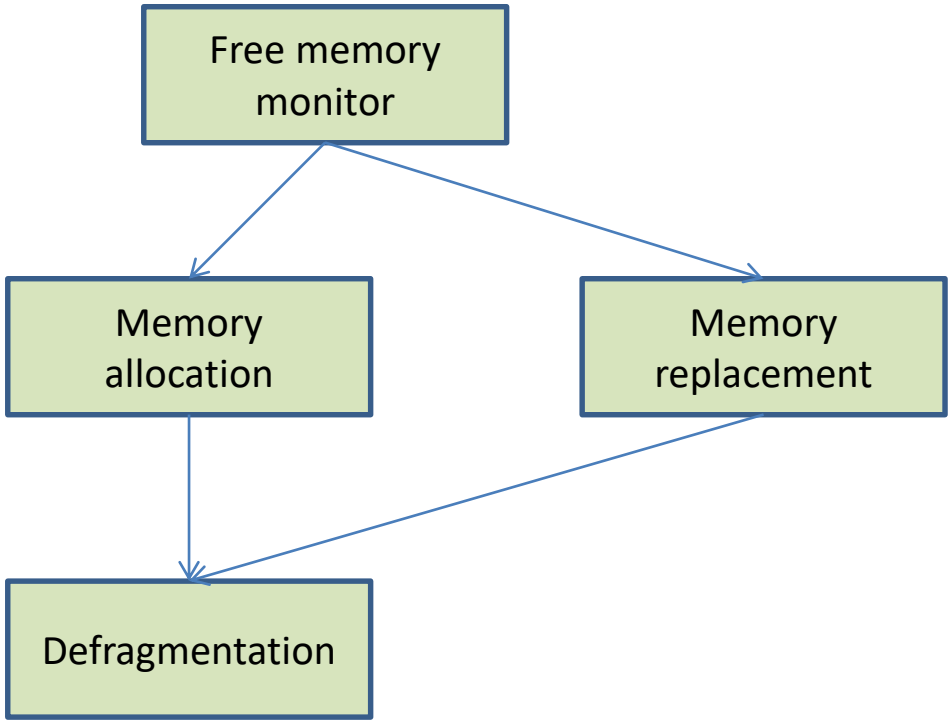
- If a process has allocated more pages than its upper threshold for the working set, it is a good candidate for page swapping. Contrary, if by taking a page the lower threshold is passed, it makes sense to swap all the pages.

- The two thresholds can be selected based on the page fault frequency. If a process generates page faults too often, it needs more pages; if it doesn't generate page faults for a time, probably it has too many pages.

Segments and pages

- Many systems work with segments that are made of pages.
- In this case, one option for over-allocation management is to simply ignore segments and work only with pages.
- Sometimes segments can contain the full working set of pages, or critical pages of the process (e.g., libraries) and they can be dealt at the segment level – swap or not the entire segment.
- However, page swapping gives more flexibility in satisfying requests.

Memory management services



Exercises and problems

1. Why Belady's Min and not frequently used are replacement techniques difficult to implement?
2. Explain the use of page status flags from the page table entry by replacement techniques.
3. A process references five pages, A,B,C,D and E in the following order: A,B,C,D,A,B,E,A,B,C,D,E. We assume initially memory is empty and FIFO is used. Find the number of page transfers.
4. A process has eight virtual pages on disk and is allocated four page frames in memory. The following page transfer into memory occurs: 1,0,2,2,1,7,6,7,0,1,2,0,3,0,4,5,1,5,2,4,5,6,7,6
   a) Show the successive pages brought into memory using LRU. Assume the frames are initially empty.
   b) Repeat a) for FIFO.

Conclusions

- Main memory is the 2nd most important resource managed by the OS. The system can use segments, pages, blocks of fix or variable size.
- The memory management services include free space registration, allocation/de-allocation, replacement, defragmentation.
- All these services can work as distinct processes or as functions of the same process. Their execution time should be kept at minimum.
- Another kernel service monitors the correct function of the main memory.