Operating systems

List 6

Exercise 1

Explain the difference between a **page** and a **frame**. Find information about **page table entries** and **page directory tables** for IA-32 architecture. Explain the meaning of each field and flag within these records. Classify flags, according to their purpose, into following sets: cache memory management, virtual memory management, access permissions, page size; operation mode.

Exercise 2

Let's consider a **page fault handler** subroutine (monolithic kernel) or a **pager** process (μ-kernel). Both of them play a role of OS subsystem, that intercepts CPU control, when a process accesses an invalid address. What data such a routine should be delivered by CPU as a result of hardware exception processing? Propose a sketch of a page fault handler subroutine (in pseudo code) that handles **demand paging** with **memory protection**. Consider secondary memory to be a black box. What extra data structures (other than page tables) have to be maintained by the kernel?

Exercise 3

Extend a subroutine from previous exercise to handle **swapping** and **memory mapped files**. For the sake of simplicity, assume that you use a single **swap partition** or **swap file**. Describe how kernel's data structures have changed.

Exercise 4

Explain a technique ubiquitous in systems with paging called **COW** (abv. *copy-on-write)*. What are the direct benefits it delivers? How is it used in practice? How to extend page fault handler routine and kernel's data structures to accommodate this technique?

Exercise 5

We already know that any memory access performed by a process involves potentially complex chain of hardware actions (i.e. employing **virtually indexed cache**, **translation lookaside buffer**, **multi-level page tables**). Discuss the shortest and the longest chain of actions, in order to estimate time of instruction execution.

Exercise 5+

Can we store page tables in memory that is amenable to paging? If we could, why would it be useful? Can you identify any difficulties with pageable page tables implementation?

Exercise 6

Let's consider four kind of memory management policies. Explain the purpose of (a) **fetch**, (b) **placement**, (c) **replacement**, (d) **cleaning policy**. We've already discussed **demand paging** technique. How does **prepaging** differ from it, why can it be useful?

Exercise 7

While discussing **cache memory**, we used **temporal locality** and **spatial locality** terms. How does locality principle apply to virtual memory? Explain following terms: **working set** and **resident set**. How locality affects these sets during process' lifetime? Find and explain a chart showing dependency between page fault frequency and page size. Why for systems with small pages, page faults occur less often, than for systems with medium sized pages?

Exercise 8

Consider a system with resident set of fixed size. We've got main memory that consist of four frames and virtual address space of eight pages. The only process that is being executed dereferences (in given order) addresses of following pages: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2. Show how following **page replacement** algorithms: `FIFO`, `LRU`, `CLOCK`, work for data above. Which one of them generates the least page faults? Please assume, that initially all frames are unused.

Exercise 9

Explain **page buffering**. Discuss advantages of that technique in context of (a) exploiting locality, (b) **working set management** and (c) **page cleaning policy**. Let's consider a system with **globally scoped variable-sized working set** management algorithm. How page buffering would affect such a system?

Exercise 10

Consider a system with demand paging. There are four processes in the system. At an arbitrary moment of time, CPU and virtual memory I/O utilization measurements were taken. Answer following questions:

- Should the OS should increase or decrease **degree of multiprogramming**?
- Was demand paging a reason of performance degradation?

… for each of measurements below:

- CPU: 13%, IO: 97%,
- CPU: 87%, IO: 3%,
- CPU: 13%, IO: 3%.

Explain **trashing** term. If an OS detected trashing, how can it choose to behave in order to lower the degree of multiprogramming?