Operating Systems

List 5

Exercises marked with plus sign are not mandatory. Though a student is encouraged to answer them, as they provide additional points not counted to max. number of points for list.

Exercise 1

Variables, as a programming language construct, are abstract (ie. symbolic) representation of certain memory locations. Function addresses are represented in similar manner. Address binding is a process that translates a symbol into certain in-memory location, that possibly exists in user address space. Give three situations where the binding can occur. Consider variable binding and function binding separately. Why can address binding be useful at run-time?

Exercise 2

Explain cursorily segmentation and paging mechanisms including address translation and auxiliary data structures (segment / page table entries). Compare both against variable-sized partition approach. List issues that were solved by use of segmentation and paging. Show advantages of variable-sized partitions over segmentation.

Exercise 3

Let's consider memory management in user space (i.e. `malloc` and `free` library routines). Assume that an OS is completely unaware of user-level allocator. Nevertheless, there has to be certain mechanism exposed by an OS, that allows to expand or shrink memory size available for the allocator. What kind of mechanism should be implemented in a system that supports: variable-sized partitions, segmentation, paging. Explain terms: external and internal fragmentation w.r.t. aforesaid memory management techniques.

Exercise 4

Explain three placement algorithms: based on free block list (`best-fit`, `first-fit`, `next-fit`), bitmap of used blocks and buddy systems. Compare them w.r.t managed block size, computational efficiency of `malloc` / `free` operations, overhead of data structure size required to hold information about blocks.

Exercise 5+

Let's assume that a process owns certain block of memory in heap area. Memory within that block is managed by a user space allocator. Apply algorithms mentioned in previous exercise in order to propose a hybrid allocator (ie. employ different techniques depending on block size). Note that programs expose certain patterns of memory allocator usage, why? You can apply additional data structures: balanced trees (e.g. `splay`, `red-black`, `b-tree`) and hash tables; to store information about unused blocks of given size efficiently.

Exercise 6

Compare two approaches for organizing data structures for paging: multi-level page table and inverted page table. Explain how address translation (from logical address space into physical one) is conveyed. Discuss size of page table required to represent a process address space. Explain briefly the meaning of extra bits in page descriptor.

Exercise 7

Consider paging with multi-level page table. What is TLB (Translation Lookaside Buffer)? Explain how does it improve efficiency of address translation mechanism. Consider how virtually indexed cache memory and TLB influence the process of address space switching, and identify issues they cause. What are huge pages and what they are used for?

Exercise 8

Let's consider a system with paging. Memory protection mechanism disallows a process to access memory that it doesn't own. However, there're situations, when a process would like to communicate with other processes by using shared memory. How to realize it in a system with paging? Note, that shared memory region can be accessible under different logical address in each process. Does it create any additional problems w.r.t. cache memories?

Exercise 9

Consider hybrid address translation technique that uses both paging and segmentation. Describe changes to address translation algorithm and data structures that desribe logical address space. Discuss advantages and disadvantages of that technique over paging or segmentation alone. Does it solve synonym problem for virtually indexed caches?

Exercise 10+

Describe briefly family of SASOS (ang. single address space operating system). Give a few examples of such OSes. Discuss their advantages over systems with private address space for each process. Show how to provide memory protection and sharing by using technique described in previous exercise.