NOTE *These weekly exercises are individual (unless marked otherwise).*

READING *chapter 9 up to and incl. 9.5.*

HAND IN *Please hand in the exercises in PDF format in Brightspace.*

Exercise 1

To understand strategies for memory management, it is useful to understand fragmentation and how to solve it.

a What kinds of fragmentation are there? Explain when they occur.

b Explain how paging is related to fragmentation, and how paging influence each kind of fragmentation.

c Depending on the viewpoint, different memory is external or internal. Situations:

1 When programming, allocating a struct with a bool, int, bool, double will insert padding between the bool and subsequent fields (3 bytes and 7 bytes), due to alignment requirements by most processor architectures.

2 If three objects are after each other in the memory, and the second object is released, it might be hard to reuse that memory for other purposes.

Related these situations to external and internal fragmentation.

Exercise 2

A 64-bit computersystem has a page size and frame size of 16 MiB (2^{24} bytes). The memory can be addressed on byte level, so the memory addresses and page table entries are 64 bits. In this exercise, we look at the efficiency of this machine.

a How large is the page table if it is at a fixed location in the RAM, so that the memory management module can always find it?

b If we page the page table, we will have a root-page table and possibly intermediate tables. Each of these intermediate tables must fit in a page of the default size, while the root table can be smaller. How many levels are needed and how large is every level (expressed in the number of items)?

c Assume that accessing the memory takes 70 nanoseconds (best case for i7 and Xeon processors, stable since 2005) and that there is a TLB (translation lookaside buffer) which makes around 80 percent of the page-lookups faster (you can assume that the TLB doesn't take time). How long does an average memory access with n root- and intermediate tables (like in part b)?

Exercise 3

A thread can access the memory of all threads within the same process. As a result a malicious thread can access the memory of the other threads as well, and as a result, bugs might occur.

a Why isn't it a problem in general that the memory of one thread is unprotected from other threads?

b Should it be allowed for processes to access each other's memory (for example, to modify a variable)? Explain why and give possible conditions.

Exercise 4

A system has N GiB memory. On this system there are processes, and each of those have two phases of execution:

1 Phase 1, which takes a long time and in which 4 GiB memory is needed

2 Phase 2, which takes a short while, but in this part, 5 GiB memory is used.

So, a process reserves 4 GiB memory once it started, and once it enters phase 2, an additional 1 GiB of memory is allocated. After these two phases, the process terminates. Answer the following questions

a Suppose, $N = 27$. Might there be a situation in which swapping is necessary to prevent a deadlock? What if the system has more/less memory? Determine conditions for N, which are sufficient for there being a deadlock. Recall: to execute a process, it must be in the memory completely.

b How many processes can this system handle if every process reserves 5 GiB immediately? Express it in terms of N.

c Can you think of a more efficient allocation method which does not swap to prevent deadlocks?