Exercise sheet 12 – Memory management

Goals:

- Memory management

Exercise 12.1: Memory management

(a) If you have a system without a MMU, is it possible to have threads?

(b) If you have a system without a MMU, is it possible to have "real" processes with all that security?

(c) What is a cache?

(d) In the context of caching: What is a hit and what is a fault?

(e) Can the operating system protect the memory of a process against others without the help of the CPU?

(f) What is position independent code (PIC)?

(g) Consider a system with virtual memory, MMU, and swapping. Is it required that the code of the executables (ELFs) is build with position independent code?

(h) Can fragmentation problems be solved by variable partition sizes (each process can choose its own required partition size)?

(i) What is swapping?

(j) Does swapping improve the performance?

(k) Is it right, that the virtual memory has to be smaller than the real memory, because the operating systems also needs some memory?

(l) What is a virtual address space?

(m) What is a page table and how is a virtual address transformed into a real address?

(n) What happens on a page fault and how is the operating system involved?

(o) Does thrashing help to improve the systems performance?

Exercise 12.2: Memory allocation strategies

Consider a main memory that contains the following free partitions: 10K, 4K, 20K, 18K, 7K, 9K, 12K, and 15K. Between the free partitions there are used partitions of an unknown size.

(a) Visualise the situation: Draw a sketch of the memory view.

Now, the following subsequent requests for memory partitions occur: 12K, 10K, and 9K.

(b) Show the results within your memory sketch when *first fit* is used.

(c) Show the results within your memory sketch when *best fit* is used.

(d) Show the results within your memory sketch when *next fit* is used.

(e) Show the results within your memory sketch when *worst fit* is used.

Exercise 12.3: Memory management programming and OS memory mechanism

(a) How and where can a process acquire (allocate) main memory in C (there are two possibilities)?

(b) How can a process release memory (distinct two possibilities)?

(c) Write a small C program that shows how the main memory acquire (allocation) and release works (distinct two possibilities).

(d) Is the operating system involved when acquire (allocation) and release of main memory is done by a process (distinct two possibilities)?

(e) Is the operating system involved when the process writes data into the main memory (distinct two possibilities)?