

# *OHJ-4106 Operating Systems*

Exam 18.3.2008

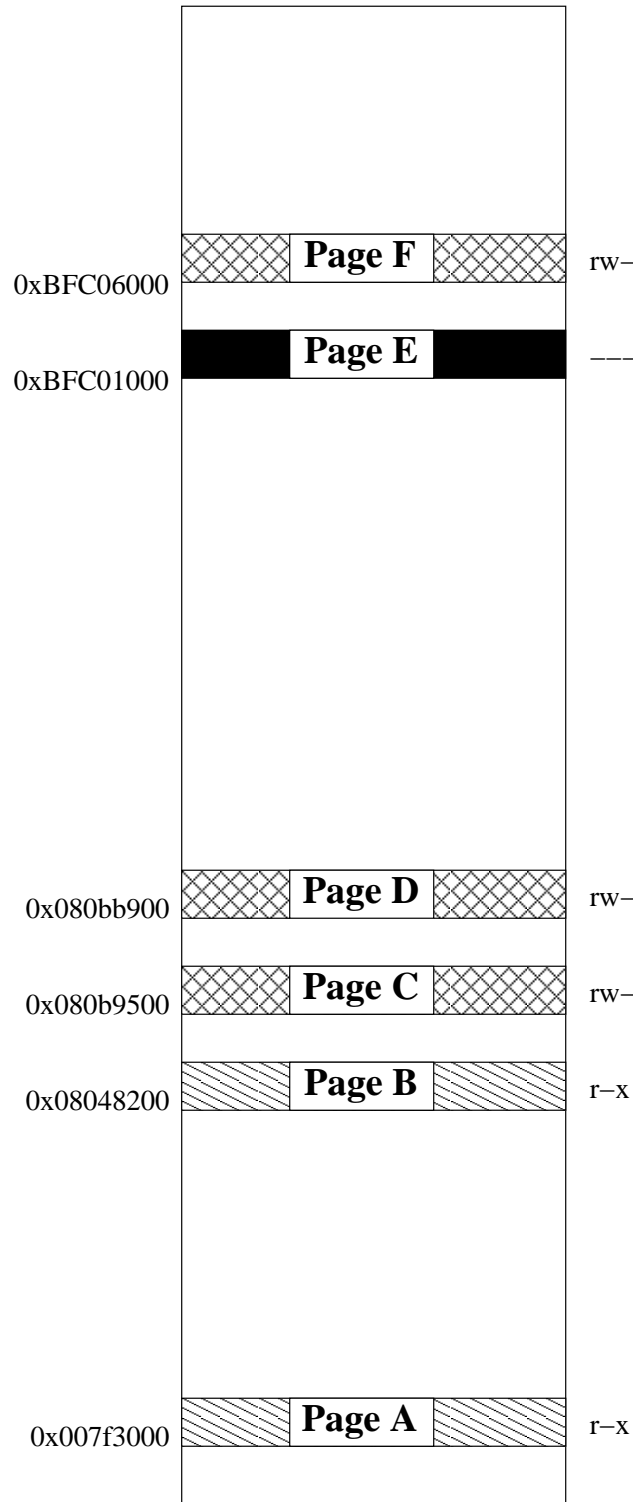
**Calculators, computers or literature are NOT allowed in this exam.**

1. Describe shortly (as to someone not familiar with this particular software systems subject) what these concepts are:
  - a) What are the main functions of an operating system?
  - b) Privileged instruction
  - c) Paging virtual memory
  - d) Context switch
  - e) *mount* operation in UFS (Unix File System).
2. A process can have the following states: RUN, READY, WAIT, SWAPPED WAIT and SWAPPED READY. Describe:
  - What is the meaning of each state?
  - What transitions are possible between the states, when do they happen and which part of the operating system typically performs the transition.
3. A newly started Linux process owns the memory pages seen in figure 1. Virtual memory addresses increase from bottom upwards and the protection bits are:
 

r	page is readable by the process
w	page is writable by the process
x	the process can execute code from the page

Explain what parts of the running process are located in each of the memory pages. Also explain where the data resides when it is not available in the main (physical) memory.
4. A user program wants to write to a file `/home/kj/log/a.txt`. What operations have to be done inside the operating system to verify that the operation can be permitted? What (how many) disk I/O operations might be needed in Unix File System to access all the necessary information? How these checks are typically optimized so that the permissions checks are not made every time the file is written into?
5. The attached listing is a part of the Minix 1.1 kernel source code. For each of the six functions describe:
  - What is the purpose of the function? (What basic operating system function does it implement?)
  - From where this function is called? (as a result of a hardware operation? some other code in the operating system or a user process?)

Also the code implements one scheduling algorithm. How does it work?



Kuva 1: Memory pages used by a process