Exam: TDIU11

Operating Systems

2022-03-25 kl: 14-18

On-call (jour): Ahmed Rezine, (Tel. 1938).

- Use the exam wrappers distributed at the exam room. Do not forget to fill each wrapper with your anonymous ID.
- Read the instructions and all the assignments before you begin answering.
- Recall that a byte is 8 bits, a KiB is 2^{10} bytes, a MiB is 2^{20} bytes, a GiB is 2^{30} bytes, a TiB is 2^{40} bytes and a PiB is 2^{50} bytes.
- You may answer in either English or Swedish. You can use a dictionary between English and another language.
- Be **precise and clearly motivate** all statements and reasoning. If in doubt about a question, write down your interpretation and assumptions.
- Write clearly. Unreadable text will be ignored!
- The exam is graded U, 3, 4, 5 (preliminary limits: 20pts, 31pts and 36pts).

Problem 1 (Processes and scheduling, 12pts)

We will schedule 3 processes P_a, P_b and P_c described in Table 1 using the multilevel feedback queue described in Figure 1.

Events	**Description**
start of tick 0	P_a arrives. It has a total burst time of 21-time units
start of tick 6	P_b arrives. It has a total burst time of 14-time units
start of tick 8	P_c arrives. It has a total burst time of 5-time units. It issues an I/O request after having executed for 3-time units. The I/O request is served, and P_c is again ready, after 2-time units.

Table 1. Events involving processes P_a, P_b and P_c

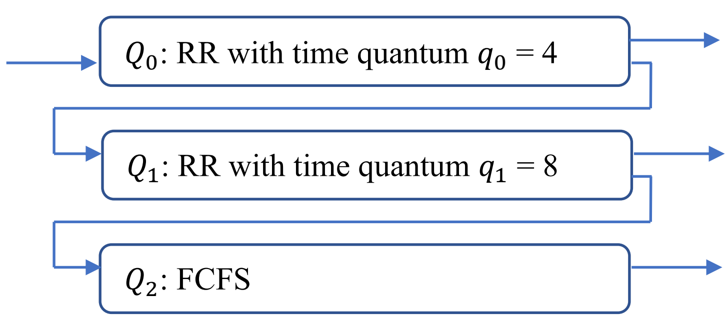

Figure 1. Multilevel feedback queue.

At their arrival, all processes are initially added to the tail (i.e., youngest in a FIFO) of the Q_0 queue. The scheduling in the multilevel feedback queue satisfies the constraints described in the following four paragraphs:

- Processes in Q_0 have the highest priority. Processes in Q_2 have the lowest priority. Processes in Q_1 are pre-empted as soon as there is a job in Q_0 that is ready to run. Processes in Q_2 are pre-empted as soon as there is a job in Q_0 or Q_1 that is ready to run. Processes in Q_1 (respectively, in Q_2) that are pre-empted because of the existence of ready processes in Q_0 (respectively in Q_0 or in Q_1) are put at the tail of Q_1 (respectively, of Q_2). In other words, running processes that are pre-empted by higher priority processes are moved to the end of their current queue.

- Processes in Q_0 are run in a Round Robin fashion. They can continue running for at most $q_0 = 4$ time units (even if new processes arrive). If pre-empted because they did not finish in $q_0 = 4$ time units, then they are moved to the tail of the Q_1 queue (hence getting a lower priority). If they release the CPU because of an I/O request, then they are moved to the tail of Q_0 (hence, staying in the same queue with the same priority).

- Processes in Q_1 are also run in a Round Robin fashion. They can continue running for at most $q_0 = 8$ time units. If pre-empted because they did not finish in $q_0 = 8$ time units, then they are moved to the tail of the Q_2 queue (hence getting a lower priority). If they release the CPU because of an I/O request, then they are moved to the tail of Q_1 (hence, staying in the same queue with the same priority).

- Processes in Q_2 are run in FCFS. If they release the CPU (e.g., because of an I/O request), then they are moved to the tail of Q_2 (hence, staying in the same queue with the same priority).

Questions:

1. Give a Gantt diagram showing which process is running from (time units 0 to 40). (3pts).

2. Give the turnaround time of each one of the three processes P_a, P_b and P_c. (2pts).

3. Give the waiting time of each one of the three processes P_a, P_b and P_c. (2pts)

4. Explain what mechanisms are involved in order to ensure that a process in queue Q_0 does not run for more than $q_0 = 4$ time units. (2pts).

5. Are there other workloads (i.e., sequences of CPU bursts of some processes) for which starvation is possible? If yes, describe such a workload (number of processes, their behaviours and burst times). If not, argue why. (3pts).

Problem 2 (Filesystem, 12pt)

Assume a filesystem with indexed allocation and where the physical blocks (i.e., physical sectors) are 512B (i.e., 2^9 bytes) large and the logical blocks are 2048B (i.e., 2^{11} bytes) large. Suppose block pointers are 32 bits large.

1. What is an inode? Give two fields you expect an inode to contain. You should pick two fields that are different from the memory indexes mentioned in question 4 of this problem. (2pts).

2. Given the size of a block pointer. How many logical blocks can be addressed? What would be the corresponding maximum disk size? (2pts)

3. Assume free blocks are tracked using a bitmap. What is the size of the bitmap if the disk size is 1 TiB (i.e., 2^{40}bytes)? (2pts)

4. What is the maximum size of a file if each inode contains 12 directly indexed blocks, 1 single-indirect block, 1 double indirect block and 1 triple indirect block. (2 pts).

5. Give one advantage and one disadvantage of indexed allocation compared to linked allocation. Explain. (2pts).

6. Give one advantage and one disadvantage of indexed allocation compared to contiguous allocation. Explain. (2pts).

Problem 3 (Memory management, 12pts)

1. What problem is solved by paging but not by segmentation? Explain. (2pts).

2. Assume a system with 12-bits virtual addresses. Physical addresses are 16-bits. Assume one-level paging with 256-bytes pages. We will consider two processes P_1 and P_2 with page tables described below. The frame numbers are given in hexadecimal, for instance, 0x10 (in hexadecimal) corresponds to 16 (in decimal) and 0xb00010000 (in binary). In the tables, the value 1 for the valid bit means the corresponding page entry is in memory and valid.

    a. In each page entry, there are 7 bits that can be used for storing information other than the frame number and the valid bit. Give examples of such information and explain when it can be used. (2pts)

    b. For each one of the following virtual addresses state whether the following virtual addresses are in main memory. If they are, state their equivalent physical address in hexadecimal. All values are given in hexadecimal:

        i. Virtual address 0x113 for process P_1 (1pt)

ii. Virtual address 0x113 for process P_2 (1pt)

iii. Virtual address 0x8F0 for process P_1 (1pt)

iv. Virtual address 0x8F0 for process P_2 (1pt)

c. Which frames do processes P_1 and P_2 share? For each such frame, give the P_1 and P_2 virtual addresses of a byte for each one of the shared frames. (4pts)

Page table of process P_1			Page table of process P_2		
Frame number	Valid bit	Other bits	Frame number	Valid bit	Other bits
0x78	0	0xb1111110	0x72	1	0xb0011111
0xC5	1	0xb0011100	0xB4	0	0xb1110011
0x2A	1	0xb0110001	0x8A	0	0xb0011110
0xFB	0	0xb1100100	0x1A	0	0xb0001011
0x5B	0	0xb1011111	0xB2	0	0xb0111100
0xC0	1	0xb1001001	0x0A	1	0xb0000001
0x8C	1	0xb0111010	0xF4	1	0xb1000110
0xD7	0	0xb1111011	0xC4	1	0xb0110111
0x45	1	0xb1001110	0xBF	1	0xb0001000
0xC4	1	0xb0111101	0xD0	0	0xb1110000
0x3C	1	0xb1100011	0x26	0	0xb0110111
0xA0	0	0xb1011101	0xCF	1	0xb1000000
0x72	0	0xb1101111	0x2A	1	0xb1010100
0x3A	0	0xb1011110	0xC0	1	0xb0110001
0x70	0	0xb1010111	0xB0	1	0xb1101010
0x0A	1	0xb1101101	0xD7	0	0xb1001100

Problem 4 (Security, 4pts)

The description in the man page of the "su" command on a Linux distribution states:

"`su allows to run commands with a substitute user and group ID. When called without arguments, su defaults to running an interactive shell as root.`"

Answer the following questions:

1. Executing "`ls -l /bin/su`" gives:
   "`-rwsr-xr-x 1 root root 67816 Feb 7 14:33 /bin/su`".
   The string "`-rwsr-xr-x`" has 10 fields. The first field states the item is a file "`-`", neither a directory "`d`" nor a link "`l`". Explain the meaning and effects of the remaining 9 flags in the string "`rwsr-xr-x`". (2pts).

2. Can it be dangerous, for this file, that the "`w`" flag appears at other positions in the string "`rwsr-xr-x`" ouput by "`ls -l /bin/su`"? If yes, describe a scenario where this could be used by an attacker, otherwise explain why not. (2pts)