https://folk.ntnu.no/michaeng/tdt4186_22/
michael.engel@ntnu.no

Theoretical exercises
Spring 2022

Theoretical Exercises 6

Realtime Scheduling, I/O Scheduling and File Systems

Please submit solutions on Blackboard by Monday, 14.3.2022 12:00h

6.1 Rate Monotonic Schedulability

The following tasks are given:

	τ_1	τ_2	τ_3
C_i	1	3	2
T_i	3	8	9

a. Test if the given task set is schedulable under RM, using the sufficient test.

b. Test if the given task set is schedulable under RM, using the necessary test.

c. Assume that the first job of each task arrives at time 0. Construct the schedule for the interval [0, 20] and illustrate it graphically. In case they exist, identify deadline misses.

6.2 Periodic Scheduling

A processor is supposed to execute the following set of tasks described by their execution times C, relative deadlines D and periods T:

	τ_1	τ_2	τ_3
C_i	2	2	4
D_i	5	4	8
T_i	6	8	12

a. Execute the sufficient schedulability test for EDF and calculate the result. What statement regarding schedulability can be made based on your result?

b. If there is a feasible schedule for the given task set, construct it graphically. Let the phase $\Phi_i = 0 \; \forall \; i$.

6.3 SSTF Disk Scheduling

Explain why SSTF scheduling tends to favor middle cylinders over the innermost and outermost cylinders.

6.4 FAT Scheduling

Requests are not usually uniformly distributed. For example, a cylinder containing the file system FAT or inodes can be expected to be accessed more frequently than a cylinder that only contains files. Suppose that you know that 50 percent of the requests are for a small, fixed number of cylinders.

a. Would any of the scheduling algorithms discussed in this chapter be particularly good for this case? Explain your answer.

b. Propose a disk scheduling algorithm that gives even better performance by taking advantage of this "hot spot" on the disk.

c. File systems typically find data blocks via an indirection table, such as a FAT in DOS or inodes in UNIX. Describe one or more ways to take advantage of this indirection to improve the disk performance.

6.5 Disk scheduling (6 points)

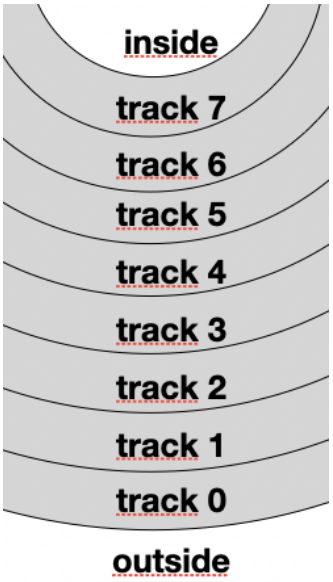
a. Assume a magnetic disk with 8 tracks. After each *second* read request (starting from L_1), the I/O scheduler receives additional read requests which are grouped (requested) together (L_2 and finally L_3).

Initially, the read/write head of the disk is at track 0.

Give the I/O scheduling order that would be performed according to the **SSTF (shortest seek time first) algorithm**:

$L_1 = \{2,4,3,1\}, L_2 = \{5,6\}, L_3 = \{0,7\}$

Access order:



b. Assume the same magnetic disk as above. The read requests in L_1 are already known to the disk scheduler. After *three* completed requests, the requests in L_2 arrive, and after three additional requests (i.e., after the sixth request), the requests in L_3 arrive.

Initially, the read/write head of the disk is at track 0.

Give the I/O scheduling order that would be performed according to the **elevator algorithm**:

$L_1 = \{1,4,7,2\}, L_2 = \{4,6,0\}, L_3 = \{5,2\}$

Access order:

6.6 FAT file system (4 points)

This question is related to a task you would need to perform if you were a computer security expert doing a *forensic analysis* of a disk.
Given a hexadecimal dump of blocks on the disk and a description of the block contents, you need to figure out the meaning of that data.
For an MS-DOS FAT16 floppy disk, you obtain the following hexadecimal dump of a *directory block* (note: addresses are in *decimal*):

```
address data bytes ..................................  ASCII representation

0009728 49 4f 20 20 20 20 20 20 53 59 53 27 00 00 00 00  IO      .SYS
0009744 00 00 00 00 00 00 08 5d 62 1b 1d 00 16 9f 00 00
0009760 4d 53 44 4f 53 20 20 20 53 59 53 27 00 00 00 00  MSDOS   .SYS
0009776 00 00 00 00 00 00 08 5d 62 1b 6d 00 38 95 00 00
0009792 43 4f 4d 4d 41 4e 44 20 43 4f 4d 20 00 00 00 00  COMMAND .COM
0009808 00 00 00 00 00 00 07 5d 62 1b b8 00 39 dd 00 00
0009824 44 42 4c 53 50 41 43 45 42 49 4e 27 00 00 00 00  DBLSPACE.BIN
0009840 00 00 00 00 00 00 08 5d 62 1b 27 01 f6 fc 00 00
0009856 4d 53 44 4f 53 20 20 20 20 20 20 28 00 00 00 00  MSDOS
0009872 00 00 00 00 00 00 1a 88 99 1c 00 00 00 00 00 00
0009888 46 44 49 53 4b 20 20 20 45 58 45 20 00 00 00 00  FDISK   .EXE
0009904 00 00 00 00 00 00 36 59 62 1b 02 00 17 73 00 00
```

The structure of a FAT16 directory entry is as follows (all numbers are stored in *little endian byte order*:

Bytes	Content
0–10	File name (8 bytes) with extension (3 bytes)
11	Attribute - a bitvector. Bit 0: read only. Bit 1: hidden. Bit 2: system file. Bit 3: volume label. Bit 4: subdirectory. Bit 5: archive. Bits 6-7: unused.
12–21	Reserved
22–23	Time (5/6/5 bits, for hour/minutes/doubleseconds)
24–25	Date (7/4/5 bits, for year-since-1980/month/day)
26–27	Starting cluster (0 for an empty file)
28-31	File size in bytes

For each directory entry, find out:

a. The name of the entry

b. The type of the entry including the set of file attributes

c. The starting cluster number

d. The file size in bytes