NTNU | Norwegian University of Science and Technology

Operating Systems

Theoretical Exercise 5: Solutions

Michael Engel

5.1 Disk scheduling

a. Assume a magnetic disk with 8 tracks. After each second read request (starting from *L1*), the I/O scheduler receives additional read requests which are grouped (requested) together (*L2* and finally *L3*).

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

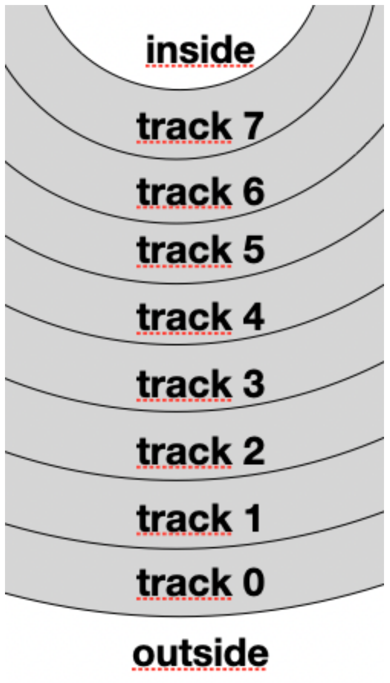
Give the I/O scheduling order that would be performed according to the **SSTF (shortest seek time first) algorithm**:

L1 = {2,4,3,1}, *L2* = {5,6}, *L3* = {0,7}

Access order:

1. L1 received and ordered SSTF: 1, 2, 3, 4
2. Read track 1 and track 2
3. L2 received, added to remaining, SSTF: 3, 4, 5, 6
4. Read tracks 3 and 4
5. L3 received, added to remaining, SSTF: 5, 6, 7, 0

=> Order: {1, 2, 3, 4, 5, 6, 7, 0}



5.1 Disk scheduling

b. Assume a magnetic disk with 8 tracks. After each ***third*** read request (starting from *L1*), the I/O scheduler receives additional read requests which are grouped (requested) together (*L2* and finally *L3*).

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**: ***that "4" was a typo, should have read "3"***

L1 = {1,4,7,2}, *L2* = {4,6,0}, *L3* = {5,2}

Access order:

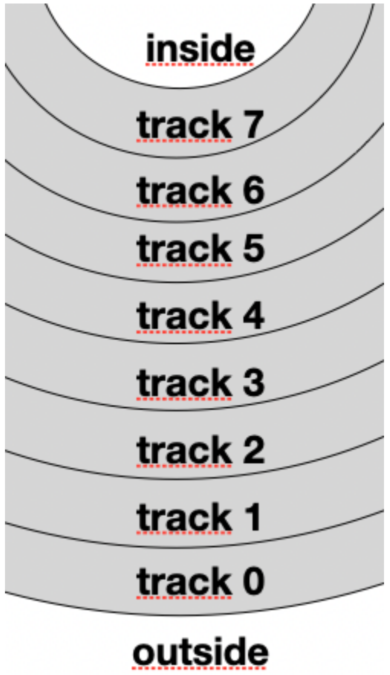
1. L1 received and ordered elevator: 1, 2, 4, 7
2. Read track 1, track 2 and track 4
3. L2 received, added to remaining, elevator: 6, 7, 0

4. Read tracks 4 ***and 4? ➛ ambiguous, 1 or 2 reads?***

➛ 4. Read tracks 4, 6 and 7!

5. L3 received, added to remaining, elevator: 5, 2, 0

=> Order: {1, 2, 4, 6, 7, 5, 2, 0}



5.1 Disk scheduling

b. Assume a magnetic disk with 8 tracks. After each ***third*** read request (starting from *L1*), the I/O scheduler receives additional read requests which are grouped (requested) together (*L2* and finally *L3*).

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:

L1 = {1,4,7,2}, *L2* = {3,6,0}, *L3* = {5,2}

with typo corrected…

Access order:

1. L1 received and ordered elevator: 1, 2, 4, 7

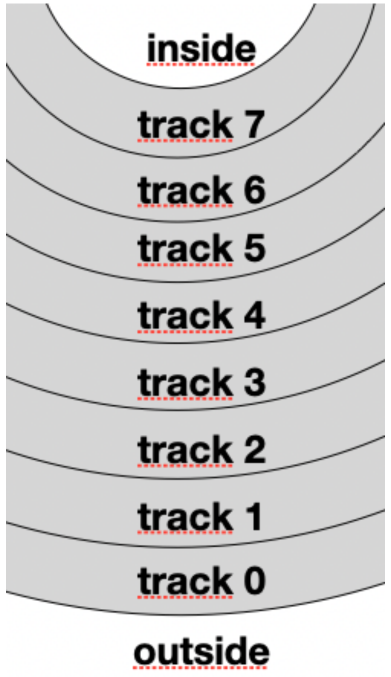
2. Read track 1 and track 2

3. L2 received, added to remaining, elevator: 3, 4, 6, 7, 0

4. Read tracks 3 and 4

5. L3 received, added to remaining, elevator: 5, 6, 7, 2, 0

=> Order: {1, 2, 3, 4, 5, 6, 7, 2, 0}



5.2 FAT file system

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.

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

5.2 FAT file system

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

That was simple enough! Note that the "." in the name is not stored on disk, but the first 8 bytes are filled with 0x20 if that part is < 8 characters!

file name | extension

```
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
```

These dots are a lie :) ***Not stored on disk***

5.2 FAT file system

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:

b. Type of the entry + file attributes

0x20 = 0010 0000 = archive

0x27 = 0010 0111 = archive, read only, system file, hidden

0x28 = 0010 1000 = archive, volume label *(so "MSDOS" is not a file)*

attribute byte

```
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 disk itself is called "MSDOS"

5.2 FAT file system

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:

c. The starting cluster number

Find bytes 26-27 (decimal)

Little endian byte order!

Cluster size on FAT12 = 512 – 4096 bytes (usually 512 on floppy disks)

```
address data bytes ................................  ASCII representation
                                   starting cluster

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  0x001d = dec. 29
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  0x006d = dec. 109
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  0x00b8 = dec. 184
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  0x0127 = dec. 295
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  0x0000 – not a file
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  0x0002 = dec. 2
```

5.2 FAT file system

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:

d. The file size in bytes

Find bytes 28-31 (decimal)

Little endian byte order!

File size is given in bytes

```
address data bytes ..................................  ASCII representation
                                                   file size
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  0x9f16 = dec. 40726
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  0x9538 = dec. 38200
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  0xdd39 = dec. 56633
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  0xfcf6 = dec. 64758
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  0x0000 – not a file
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  0x7317 = dec. 29463
```