Operating Systems Exercise 4 Deadline 12.10.09

1. A serial port receives one byte at a time over a serial data line. When designing the driver, it was considered wether to use active waiting or interrupt-based reading of incoming data.

   a) If the processing of interrupts (the interrupt routine, as well as all the extra time spent activating the interrupt functinality) takes 500 ns, at which bit-rate (bits/second) does active waiting become the better choice? Ignore the time it takes to transfer the data from the serial port to memory.
   b) Generally, a hardware buffer of size 16 bytes is used to temporarily buffer incoming data. How would such a buffer affect the choice between active waiting and interrupt-based reading?
   (2p)

2. SSD (Solid State Disk) is gaining popularity on the market. There are two kinds of SSD, SLC and MLC. Search the web for information regarding the differences between SLC and MLC technology. Is one preferable over the other when used for the following: (2p)
   a) As the system drive of an operating system.
   a) As storage for multimedia (Video / Audio).
   a) Storing a database.

3. Requests arrive to the hard disk driver for cylinders 20, 7, 52, 2, 18, 42, and 28, in the given order. A search will take 8 ms for each cylinder that the arm has to move over (i.e. moving the arm from cylinder 20 to 22 takes 16 ms). How long is the total search time, when using the following disk arm scheduling algorithms

   a) First come, first served
   b) Shortest seek first
   c) Elevator algorithm

   Assume that the arm is positioned at cylinder 20 initially. (2p)

4. The data structures used by the ext2 filesystem in linux are defined in the file */usr/include/linux/ext2_fs.h.* The following lines can be found in said file:

```
.....
#define EXT2_NDIR_BLOCKS          12
#define EXT2_IND_BLOCK            EXT2_NDIR_BLOCKS
#define EXT2_DIND_BLOCK           (EXT2_IND_BLOCK + 1)
#define EXT2_TIND_BLOCK           (EXT2_DIND_BLOCK + 1)
#define EXT2_N_BLOCKS             (EXT2_TIND_BLOCK + 1)
...
__le32  i_block[EXT2_N_BLOCKS];/* Pointers to blocks */'
...
```

   where the constant EXT2_NDIR_BLOCKS defines how many blocks are directly addressed in the i-node. The constants EXT2_IND_BLOCK, EXT2_DIND_BLOCK and EXT2_TIND_BLOCK give indices for block addresses using single, double and triple indirection, respectively. The addresses used are of size 32 bits. What is the largest possible file size in the file system if using
   a) 1 kB blocks

b) 4 kB blocks
(2p)

5. Using the command *mkisofs* (available at for example lenz.cs.abo.fi, if you want to try it), one can build an ISO9660 file system. An example of such a file system can be downloaded from http://www.abo.fi/~jbjorkqv/opsys_09/test.iso

An ISO9660 file system consists of 2048-byte sectors. The first 16 sectors are reserved for future use, and consist of zeros. Sector 17 contains the *primary volume descriptor*, where the first bytes are structured as described by *struct t_isovoldesc* in the code below.
Download test.iso. Compile and test the code. Then add code to print sys_ident, and the total amount of sectors, as well as the size of the file system in bytes. Describe your code and attach a test printout. (2p)

```
#include <stdio.h>
#define FILENAME "test.iso"
char volident[] = {67,68,48,48,49,1};
struct  t_isovoldesc {
  char start;
  char desc[7];
  char sys_ident[32];
  char vol_ident[32];
  char zeros[8];
  long long sectors;
};
struct t_iso_sector {
  char data[2048];
};
int main() {
  FILE *fd;
  struct t_iso_sector iso_sector;
  struct t_isovoldesc *isovoldesc;
  fd = fopen(FILENAME, "r");
  while (!feof(fd)) {
    fread(&iso_sector, sizeof(iso_sector), 1, fd);
    isovoldesc = (struct t_isovoldesc *)&iso_sector;
    /* Search for sector containing volume descriptor */
    if (strcmp(isovoldesc->desc, volident)==0) break;
  }
  printf("Volume name: %s\n", isovoldesc->vol_ident);
  return 0;
}
```