OPERATIVSYSTEM 2013, ÖVNING 4, 28.11.2013, deadline 13.12.2013

1. Seriekommunikation används som grund i de flesta former av kommunikation mellan komponenter och datorer (RS-232, RS-485, SPI, USB, CAN, Ethernet...). Det som vi tänker på när vi hör seriekommunikation är ändå de enklare modellerna, type RS-232 och RS-485. (2p)

   Vid seriekommunikation finns det olika sätt att hantera seriekommunikationen:

   a) Den hårdvarumässigt enklaste, men mjukvarumässigt svåraste är att direkt driva i/o-portar. För att läsa en asynkron serieport (dvs. avsändare och mottagare har ingen genensam klocksignal) måste man sampla 3 x baud-hastigheten. Om man använder ett timer-avbrott för att sampla, och timer-avbrotthanteringen tar 500 ns (inklusive sampling) att utföra, vilken är den maximala baud-hastigheten vid mottagning?
   b) Det normala sättet är att det finns hårdvarustöd för asynkron seriekommunikation, UART (Universal Asynchronous Receiver Transmitter), typiskt med bufferstorlek 1 eller 16 byte, där en inkommen byte på en serieport orsakar ett avbrott. Om avbrottshanteringen fortfarande tar 500 ns, vilken är maximala baud-hastigheten vid mottagning i fallet med 1 resp. 16 byte buffer?

2. SSD (Solid State Disk) börjar bli allt vanligare i bärbara datorer. SSD finns i två varianter, SLC och MLC. Sök på nätet skillnaden mellan SSD gjort med SLC och MLC teknik, sök också prisuppgifter. Är någondera att föredra (att varför), eller lönar det sig med vanlig magnetisk hårdskiva, för följande: (2p)

   a) Som systemskiva för operativsystem
   b) Lagring för multimediamaterial (Video / Audio)
   c) Lagring av databas

   Uppgör tabell, med följande layout, och fyll i fördelar / nackdelar

	Pris (€/MB)		Systemskiva	Multimedia	Databas
SSD SLC		Fördel			
		Nackdel			
SSD MLC		Fördel			
		Nackdel			
Magnetisk skiva		Fördel			
		Nackdel			

3. Uppgör ett program som statistiskt analyserar följande rutiner för hårdskivearmskedulering. (4p)
   - FCFS (First Come, First Served)
   - Närmaste cylinder nästa
   - Hissalgoritmen

   Programmet kan uppgöras på följande sätt:
   a) En räcka av hårdskiveaccessförfrågningar (1000 kan vara lämpligt) skapas slumpmässigt (antag att det finns 1000 cylindrar på hårdskivan)
   b) De olika rutinerna ovan kallas en åt gången, statistik uppgörs för medelsöktid, om vi antar att det tar 50 ns per cylinder för hårdskivearmen att flytta sig

4. Datastrukturer för filsystemet ext2 är i linux definierade i filen */usr/include/linux/ext2_fs.h*. Nedan är några rader ur filen:

.....

```
#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 */'
...
```

där konstanten *EXT2_NDIR_BLOCKS* anger hur många block man direkt addresserar från i-noden. Konstanterna *EXT2_IND_BLOCK*, *EXT_DIND_BLOCK* resp. *EXT2_TIND_BLOCK* anger index för enkel, dubbel respektive trippel indirektion till block-adresser. Block-adresser är 32 bit. Om block-storleken är a) 1kB b) 4kB, hur stor är den största filstorleken i filsystemet? (2p)

5. Med kommandot *mkisofs* (finns på t.ex. lenz.cs.abo.fi) kan man bygga upp ett ISO9660-filsystem (används typisk på CD-ROM / DVD). Ett exempel på ett sådant filsystem finns på

   http://www.abo.fi/~jbjorkqv/opsys_12/test.iso

   Ett ISO9660 filsystem består av sektorer på 2048 byte. De första 16 sektorerna (0-15) bestå av nollor (de är reserverade för framtida bruk). Sektor 17 innehåller *primary volume descriptor*, med format som i början på *struct t_isovoldesc* i koden nedan. Ladda ner *test.iso*, kompilera och testa koden. Lägg därefter till kod som dessutom skriver ut sys_ident samt totala antalet sektorer samt storleken på hela filsystem i bytes. Beskriv koden och bifoga resultat. (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 (strncmp(isovoldesc->desc, volident,6)==0) break;
  }
  printf("Volume name: %s\n", isovoldesc->vol_ident);
  return 0;
}
```