OPERATIVSYSTEM 2013, ÖVNINGSUPPGIFT 1, 8.11.2013
OBS. Räkneövning inlämnas elektronisk på adressen https://abacus.cs.abo.fi/ro.nsf.
Deadline: Fredag 15.11.2013 kl. 16.00. Wictor Lund håller i trådarna för övningarna. Wictor har meddelat att han vill ha väl ifyllda övningsrapporter!

Datorn *lenz.cs.abo.fi* kan användas för övningsuppgifter (finns bl.a. *strace* installerat). För att få tillgång till datorn, måste man få ett hemområde. Skicka epost till Wictor (wlund at abo.fi) så fixar det sig.

1. Med kommandot `strace` (finns i de många linux-distributioner) kan man lista alla de anrop som görs till kärnan. Kompilera och exekvera HelloWorld-exemplet (nedan). Räkna upp alla systemanrop som görs, samt frekvensen av dessa. (1p)

```
// File helloworld.c
#include <stdio.h>

int main() {
  printf("Hello World\n");
}
// ------------------

Kompilering och körning
% gcc helloworld.c –o helloworld
% ./helloworld
% strace ./helloworld
```

2. I koden ”`iotest.c`” (som bilaga nedan) finns två versioner implementerade för skrivande till fil, en användande av `fwrite() (i libciotest())`, en annan av `write()(i rawciotest())`. Kompilera koden och kör den, med användande av turvis endera implementationen. Använd hjälpprogrammet `time` (används `time <program som skall timas>`) för att mäta körtiden för respektive implementation. Experimentera med olika antal iterationer för att skriva till en fil. Rapporterna resultaten. Du borde se skillander i tid mellan de olika metoderna, vad beror skillanderna på? (3p)

3. Ett soft-realtidssystem har fyra periodiska händelser med perioder om 50, 100, *y* och 250 ms, där *y* är 10 * dag i månaden för ditt födelsedatum (t.ex *y*=170 för födelsedag 17.7). Anta att de fyra händelserna kräver 35, 20, 10 och *x* ms processortid respektive. Vilket är det största värde på *x* för vilket systemet är skedulerbart?

    a. Beräkna och motivera. (2p)
    b. Provimplementera på FreeRTOS. Vad händer när x blir för stort?

4. På nästa sida ges ett program som delar sig i två processer. Faderprocessen läser från tangentbordet och matar till ett rör (pipe). Dotterprocessen läser från röret och skriver ut på skärmen. Modifiera programmet så att en tredje process behandlar strängen, så att den gör en palindrom av strängen (dvs. inverterar bokstavsordningen). Kommunikationen sköts fortsättningsvis med rör. (4p)

```c
      /* pipes.c
 * compile using: gcc -o pipes pipes.c
 * run using ./pipes
 */

#include <stdio.h>
#include <unistd.h>

#define BUFFERSIZE 100

int main() {

  int pid;
  char buffer[BUFFERSIZE]; /* Allocate buffer for  temporary data */
  int files[2];  /* File descriptor in unix is a integer, used
                    together with operation read, write */

  /* Create the pipe, files[0] and files[1] will contain
     the file descriptors of both ends, note, writing will happen
     to files[1], reads from files[0] */
  pipe(files);

  /* Create the child process */
  if ((pid = fork())) {
    /* OK, I'm the parent */
    while (1) {
      printf("Parent; Enter string to put into the pipe: ");
      /* Read from the standard input (=keyboard...) */
      gets(buffer);

      /* Write the complete buffer to the pipe */
      write(files[1], buffer, BUFFERSIZE);
      sleep(1);
    }
  } else {
    /* Ok, I'm the child */
    while(1) {
      /* Read from the pipe, read the complete buffer */
      read(files[0], buffer, BUFFERSIZE);
      /* Output what was read ... */
      printf("Child: From pipe; '%s'\n", buffer);
    }
  }
}
```

```c
/* ************
  iotest.c
 *************/
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define FILENAME "testfile.txt"
#define STRLEN 50
#define NUMWRITES 10000

char *getrandstring() {
```

```
  int i;
  static char buf[STRLEN];

  for (i=0; i<STRLEN; i++) {
    buf[i] = (rand() % 28) +65;
  }
  return buf;
}

int libciotest() {
  FILE *fp;
  int i;
 // Open file
  fp = fopen(FILENAME, "w");
  if (!fp) {
    printf("Could not open %s\n", FILENAME);
    return -1;
  }
  // DO writes
  for (i=0; i<NUMWRITES; i++) {
    fwrite(getrandstring(), STRLEN, 1, fp);
  }
}

int rawiotest() {
  int fd;
  int i;

  fd = open(FILENAME, O_WRONLY);
  if (fd==-1) {
    printf("Could not open %s\n", FILENAME);
    return -1;
  }
  // DO writes
  for (i=0; i<NUMWRITES; i++) {
    write(fd, getrandstring(), STRLEN);
  }
  close(fd);
}

int main() {
  //libciotest();
  //rawiotest();
}
```