תכנות מתקדם

תרגיל תכנות מספר 1

בעזרת **סגמנטי זיכרון משותפים וסמפורים או busy wait**, עליך לממש מנגנון המכניס ממד מסוים של אקראיות בהקצאת תהליך למטרה מסוימת.

עליך לממש רוטינה

```
void assign_random_pid(char path[],

        char name[], char arg[], int n)
```

המקצה תהליך להרצת תוכנית שמסלול ההגעה עליו הוא path, השם שינתן לו הוא name ופרמטר יחיד arg, כאשר התהליך שיבחר יהיה אחד מתוך n תהליכים שה-n % pid שלי יהיה מקסימלי. כל התהליכים "מדווחים" על ה- pid שלהם למשאב זיכרון משותף, כולם חוץ מהאחרון ממתינים והאחרון המדווחים ישחרר את כל הממתינים. התהליך עם השארית הגדול ביותר ישנה את הקוד שלו לקוד שהמסלול שלו מתואר ב-path. היתר מסיימים.

לדוגמא, אם זה הקוד של קובץ ביצועי winner:

```
int main(int argc, char *argv[])

{


 if(argc > 1)

  printf("arg = %s\n", argv[1]);

 printf("I am process %d\n", getpid());


}  // main
```

אזי הפלט של התוכנית הבאה:

```
int main()
{
```

```
  int id, *nptr;
  int memid;

assign_random_pid("./winner",

        "winner", "arg1", 7);

} // main
```

יהיה:

```
 pid = 28009

pid = 28010

pid = 28011

pid = 28012

pid = 28013

pid = 28014

pid = 28015

arg = arg1

I am process 28013
```

```
#include<stdio.h<
#include<stdlib.h<
#include<sys/types.h<
#include<sys/shm.h<
#include<unistd.h<

void assign_random_pid(char[], char[],char[],int);

int main}()

      assign_random_pid("./winner","winner","arg1",7);

{

void assign_random_pid(char path[], char name[],char arg[],int
n)}

      int id, *nptr, memid;
      int newN=n+1,i,j,maxid,*nptr2,memid2;
      struct shmid_ds buff;

      //will allocate new shared area
      if((memid=shmget(0,sizeof(int),0666|IPC_CREAT|IPC_EXCL))<0)
      perror("cannot shmget!");
      if((nptr=(int*)shmat(memid,0,0))==(int*)-1)
      perror("cannot shmat!");
      if((memid2=shmget(0,sizeof(int),0666|IPC_CREAT|IPC_EXCL))<0)
      perror("cannot shmget!");
      if((nptr2=(int*)shmat(memid2,0,0))==(int*)-1)
      perror("cannot shmat!");

      *nptr=0;
      *nptr2=0;
      //create n processes
      for(i=0; i<newN; i++)}
            //if the last process free all
            if(i==n)}
                  *nptr2=1;
                  exit;(0)
            {
            //else create new process and put them to busy wait
state
            else}
                  id=fork;()
                  if(id==0)}
                        printf("pid = %d\n",getpid());
                        if((getpid()%n)>((*nptr)%n))}
                              *nptr=getpid;()
                        {
                        while}(1)
                              if(*nptr2==1)}
                                    if(getpid()==*nptr)}
                                          if(shmdt(nptr)<0)
                                                perror("cannot
shmdt");

      if(shmctl(memid,IPC_RMID,&buff)<0)
```

```
                                                        perror("cannot
shmctl");
                                                  if(shmdt(nptr2)<0)
                                                        perror("cannot
shmdt");

      if(shmctl(memid2,IPC_RMID,&buff)<0)
                                                        perror("cannot
shmctl");
                                                  execv(path,(char
*[]){name,arg,NULL});
                                                  printf("Winner is
%d\n",*nptr);
                                            {
                                            exit;(0)
                                      {
                                {
                          {
                    {
            sleep;(1)
      {
{
```

תכנות מתקדם

תרגיל מספר 2

עליך לממש רוטינה

```
typedef void (*FUN_PTR)();


void menuloop(char menu_string[], int n, FUN_PTR farr[] )
```

המקבלת מחרוזת ו-n פוינטרים לפונקציה farr וממשת תפריט שבו החזרה לתוכנית הקוראת יהיה ע"י באמצעות \-ctrl וחזרה לתפריט יהיה באמצעות ctrl-C.

במידה ואחד הפוינטרים לפונקציה שגוי, הטיפול יהיה הדפסת הודעה מתאימה אבל אחרת זהה ל-Ctrl-C. הטיפול במקרה של פוינטר שגוי ו-Ctrl-C חייב להיות ברוטינה משותפת.

לדוגמא,

פלט אפשרי של התוכנית הבאה:

```
void ha_ha_loop()
{
  while(1)
    {
      puts("Ha  Ha  Ha ");
      sleep(3);
    } /* while */

} /* ha_ha_loop */

void dollar_loop()
{
  while(1)
   {
     puts("$$$$$$$$$$ ");
     sleep(3);
   } /* while */
} /* dollar_loop */


void doubt_loop()
{
  while(1)
   {
```

```
      puts("?????????? ");
      sleep(3);
    } /* while */


} /* dollar_loop */



void menuloop(char menu_string[], int n, FUN_PTR farr[] )
{
  char answer[80];
  int k;

   if (signal(SIGQUIT, jumper2) == SIG_ERR)
       syserr("signal");

   if (sigsetjmp(jmpbuf2,1) != 0)
     {
      puts("Returning ...");
      return;
     }// if


  while (1)
  {
   if (signal(SIGINT, jumper1) == SIG_ERR)
      syserr("signal");
  if (signal(SIGSEGV, jumper1) == SIG_ERR)
       syserr("signal");

   if (sigsetjmp(jmpbuf1,1) != 0)
     puts("... Returning to menu.");

    puts(menu_string);

    fgets(answer, 80, stdin);

    k = answer[0] - '0' - 1;
   (*farr[k])();

  } /* while */

} /* mainloop */



int main()
{
  char menu_string[80];
```

```
  FUN_PTR farr[3];

 strcpy(menu_string,"Press 1 for Ha Ha Ha.\n");
 strcat(menu_string,"Press 2 for $$$$$$$$.\n");

 strcat(menu_string,"Press 3 for ????????\n");
 strcat(menu_string,"Press 4 for Error\n");

 farr[0] = ha_ha_loop;
 farr[1] = dollar_loop;
 farr[2] = doubt_loop;
 farr[3] = NULL;

 menuloop(menu_string, 4, farr);
 return 0;

} /* main */
```

יהיה:

```
Press 1 for Ha Ha Ha.
Press 2 for $$$$$$$$.
Press 3 for ????????
Press 4 for Error

1
Ha  Ha  Ha
Ha  Ha  Ha
^CKeyboard Interrupt
... Returning to menu.
Press 1 for Ha Ha Ha.
Press 2 for $$$$$$$$.
Press 3 for ????????
Press 4 for Error

4
Invalid Function pointer
... Returning to menu.
Press 1 for Ha Ha Ha.
Press 2 for $$$$$$$$.
Press 3 for ????????
Press 4 for Error

2
$$$$$$$$$$
^CKeyboard Interrupt
... Returning to menu.
Press 1 for Ha Ha Ha.
Press 2 for $$$$$$$$.
```

```
Press 3 for ????????
Press 4 for Error

4
Invalid Function pointer
... Returning to menu.
Press 1 for Ha Ha Ha.
Press 2 for $$$$$$$$$.
Press 3 for ????????
Press 4 for Error
Press 1 for Ha Ha Ha.
Press 2 for $$$$$$$$$.
Press 3 for ????????
Press 4 for Error

3
??????????
??????????
??????????
^CKeyboard Interrupt
... Returning to menu.
Press 1 for Ha Ha Ha.
Press 2 for $$$$$$$$$.
Press 3 for ????????
Press 4 for Error

4
Invalid Function pointer
... Returning to menu.
Press 1 for Ha Ha Ha.
Press 2 for $$$$$$$$$.
Press 3 for ????????
Press 4 for Error

^\Returning ...
```

```
#include <stdio.h<
#include <setjmp.h<
#include <signal.h<
#include <string.h<

static sigjmp_buf jmpbuf1;
static sigjmp_buf jmpbuf2;
typedef void(*FUN_PTR);()

void ha_ha_loop}()
      while}(1)
            puts("Ha Ha Ha");
            sleep;(3)
      {
{

void dollar_loop}()
      while}(1)
            puts;("$$$$$$$$$$")
            sleep;(3)
      {
{

void doubt_loop}()
      while}(1)
            puts;("??????????")
            sleep;(3)
      {
{



void jumper1(int signo)}
      if(signo==SIGINT)}
            puts("Keyboard Interrupt");
      {
      if(signo==SIGSEGV)}
            puts("Invalid Function pointer");
      {
      siglongjmp(jmpbuf1,1);
      perror("Longjmp return");
{

void jumper2}()
      siglongjmp(jmpbuf2,1);
      perror("Longjmp return");
{

void menuloop(char menu_string[], int n, FUN_PTR farr[] )}
      char answer;[80]
      int k;
      if (signal(SIGQUIT, jumper2) == SIG_ERR) /*ctrl/*\-
            perror("signal");
      if (sigsetjmp(jmpbuf2,1) != 0)}
            puts("Returning ...");
            return;
       //{if
```

```
      while}(1)
            if (signal(SIGINT, jumper1) == SIG_ERR) /*ctrl-C/*
                  perror("signal");
            if (signal(SIGSEGV, jumper1) == SIG_ERR) /*functions
ptr error/*
                  perror("signal");
            if (sigsetjmp(jmpbuf1,1) != 0)
                  puts("... Returning to menu.");
            puts(menu_string);
            fgets(answer, 80, stdin);
            k = answer[0] - '0' - 1;
            (*farr[k]);()
       */ {while/*
 */ {menuloop/*

int main}()
      char menu_string;[80]
      FUN_PTR farr;[3]
      strcpy(menu_string,"Press 1 for Ha Ha Ha.\n");
      strcat(menu_string,"Press 2 for $$$$$$$$.\n");
      strcat(menu_string,"Press 3 for ????????\n");
      strcat(menu_string,"Press 4 for Error\n");
      farr[0] = ha_ha_loop;
      farr[1] = dollar_loop;
      farr[2] = doubt_loop;
      farr[3] = NULL;
      menuloop(menu_string, 4, farr);
      return 0;
 */ {main/*
```

תכנות מתקדם

תרגיל מספר 3

במערכת Linux יש תוכנית שירות /usr/bin/ispell המקבלת קובץ טקסט ובודקת איות שלהם באנגלית. אם יש צורך תוכלו להתקין אותו בפקודה

apt-get install ispell

או

sudo apt-get install ispell

אם זה לא מצליח תריצו בתור root את firefox ותיכנסו

http://packages.ubuntu.com/precise/ispell

תורידו ותתקינו ידנית את ispell.

תלחצו amd64 ואחר כך אחד מרשימות ה-mirrors.

מהאתר

http://packages.ubuntu.com/trusty/iamerican

תתקינו ידנית את המילון.

תלחצו all ואחר כך אחד מרשימות ה-mirrors.

כאשר ispell מורץ עם אופציה a– הוא מסכים להיות פילטר. לדוגמא, עם הוא מורץ בצורה על קלט מהמקלדת של

hhh

fg

ggg

above

therefore

nnghg

ccd

supose

gg

הפלט שלו יהיה:

@(#) International Ispell Version 3.3.02 12 Jun 2005

& hhh 2 0: hah, huh

& fg 14 0: Ag, f, fa, fag, Fe, fig, Fm, fog, Fr, ft, g, Hg, kg, Mg

& ggg 4 0: egg, gag, gig, Gog

*

*

nnghg 0

& ccd 11 0: BCD, cad, Cd, CDC, Cid, cod, cud, c Cd, c-Cd, LCD, uCD

& supose 1 0: suppose

& gg 16 0: Ag, egg, g, Ga, gag, gb, Gd, Ge, gig, GM, go, Gog, gs, Hg, kg, Mg

כלומר הוא ידפיס * כשר המילה תקינה,

או & כאשר לא, תלוי אם יש לו אלטרנטיבות חוקיות.

1. עליך לממש רוטינה filter.c המסננת מהפלט של ispell –a רק את המילים הלא חוקיות עצמן. מותר לך לקרוא ולכתוב רק בעזרת קריאות המערכת read ו-write.
2. בעזרת cat השופכת קובץ למסך ו-filter עליך לממש תוכנית המדפיסה מקובץ w.txt את המילים הלא חוקיות (לא תוספות כלשהן).

לדוגמא, הפלט של התוכנית שלך מקובץ w.txt עם התוכן שלעיל יהיה:

```
hhh
fg
ggg
ccd
supose
gg
```

```
#include <stdio.h<
#include <unistd.h<
#include <errno.h<
#include <sys/types.h<
#include <stdlib.h<
#include <fcntl.h<
#include <sys/stat.h<
#include <sys/wait.h<

void syserr(char str[])
}
  perror(str);
  exit;(1)
 */ {sys_arr/*

int main()
}
  int  pfd;[2]
  int status;


  if (pipe(pfd) == -1)
    syserr("pipe");


  switch(fork())
}
   case -1:
     syserr("fork");
   case 0:
     if (dup2(pfd[1], 1) == -1)
         syserr("dup2");
     if (close(pfd[0])  == -1 || close(pfd[1])  == -1)
        syserr("close");
     execlp("ispell", "ispell", "-a", NULL);
     syserr("execlp1");
 */ {  switch/*
  switch(fork())
}
   case -1:
     syserr("fork");
   case 0:
}
     if (dup2(pfd[0], 0) == -1)
         syserr("dup2");
     if (close(pfd[0])  == -1 || close(pfd[1])  == -1)
        syserr("close");
     execlp("./filter", "filter", NULL);
     syserr("execlp2");
{
 */ {  switch/*
if (close(pfd[0])  == -1 || close(pfd[1])  == -1)
     syserr("close");
  while (wait(&status) != -1)
;
  return 0;
 */ {main/*
```

```
 */pipe5.c - implement date | wc/*  .

#include <stdio.h<
#include <unistd.h<
#include <errno.h<
#include <sys/types.h<
#include <stdlib.h<
#include <fcntl.h<
#include <sys/stat.h<

void syserr(char str[])
}
  perror(str);
  exit;(1)
 */ {sys_arr/*


void main()
}
      int i = 0, fd;
      char msg;[2000]

      do}
            if(read(0, &msg[i], 1) == -1)
                  syserr("read");
            i;++
      {while(msg[i-1] != '\0;('

      if((fd = open("w.txt", O_RDWR|O_TRUNC)) == -1)
            syserr("open");
      i = 0;

      while(msg[i] != '\0')
      }
            if(msg[i] == '#' || msg[i] == '&')
            }
                  i += 2;
                  while(msg[i] != ' ')
                  }
                        write(fd, &msg[i], 1);
                        i;++
                  {
                  msg[i] = '\n;'
                  write(fd, &msg[i], 1);
            {
            i;++
      {

      system("cat w.txt");
      close(fd);
      exit;(0)
{
```

תכנות מתקדם

תרגיל תכנות מספר 4

נתון האלגוריתם הבא לבדיקת ראשוניות:

בהינתן מספר n (אי זוגי למען הפשטות) לבדיקת ראשוניות, מקצים מערך flags של n+1 בתים.

מאתחלים את המערך כולו ב-1.

סורקים את המערך מ-3 עד שורש n:

עבור כל i שעבורו flags[i] == 1 מאפסים את הכניסות ב- flags באינדקסים הכפולות של i כלומר flags[2*i], flags[3*i], flags[4*i], flags[5*i] וכו'.

אם בשלב כלשהוא מאפסים את flags[n] המספר אינו ראשוני, עוצרים ומדווחים.
המספר שהכפולה שלו איפסה את flag[n] מחלק את n.

אם אחרי כל האיפוסים flags[n]==1 המספר ראשוני.

כתוב תוכנית ראשית הקוראת מספר n מהמשתמש המחלקת את מלאכת האיפוסים בין 2 thread-ים.

חלק את האיפוסים כך ש:

- ה-thread הראשון יאפס כפולות של i=3,7,11,..
- ה-thread הראשון יאפס כפולות של i=5,9,13,..
- מערך הדגלים חייב להיות משותף לשניהם.
- אם אחד ה-thread-ים מגלה שהמספר אינו ראשוני, שני ה-thread -ים חייבים לעצור.

לדוגמא, פלט אפשרי של התוכנית שלך יהיה כזה:

Enter n:

100000003

100000003 is NOT a prime.

100000003 / 643 = 155521

```
#include <stdio.h<
#include <stdlib.h<
#include <pthread.h<
#include <math.h<

pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;

void *th3;()
void *th5;()

int *arr;
long n = 2, sqrn, answer;

void main()
}
   pthread_t thread1, thread2;
   long i;

   while(n%(long)2 == 0)
}
      printf("\nEnter odd number:\n");
      scanf("%ld", &n);
{

   if(!(arr = (int*)malloc((n + 1)*sizeof(int))))
}
      printf("malloc");
      exit;(0)
{

   for(i = 0; i < n + 1; i++)
      arr[i] = 1;

   sqrn = (long)sqrtl((long double)n);
   pthread_create(&thread1, NULL, &th3, NULL);
   pthread_create(&thread2, NULL, &th5, NULL);
   pthread_join(thread1, NULL);
   pthread_join(thread2, NULL);

   if(arr[n] == 1)
      printf("\n%ld is a PRIME number!\n", n);
   else
      printf("\n%ld is NOT a PRIME number!\n%ld / %ld = %ld\n", n,
n, answer, (n/answer));
{

void *th3()
}
      long i = 3, j;

      while((arr[n] != 0) && (i <= sqrn))
      }
            j = i;
            while(j < n + 1)
            }
                  if(arr[j] == 1)
                  }
                        pthread_mutex_lock(&mutex1);
```

```
                        if(arr[n] == 0)
                        }
                              pthread_mutex_unlock(&mutex1);
                              return NULL;
                        {
                        arr[j] = 0;
                        if(j == n)
                        }
                              answer = i;
                              pthread_mutex_unlock(&mutex1);
                              return NULL;
                        {
                        pthread_mutex_unlock(&mutex1);
                  {
                  if(arr[n] == 0)
                        return NULL;
                  j += i;
            {
            i += 4;
      {
      return NULL;
{

void *th5()
}
      long i = 5, j;

      while((arr[n] != 0) && (i <= sqrn))
      }
            j = i;

            while(j < n + 1)
            }
                  if(arr[j] == 1)
                  }
                        pthread_mutex_lock(&mutex1);
                        if(arr[n] == 0)
                        }
                              pthread_mutex_unlock(&mutex1);
                              return NULL;
                        {
                        arr[j] = 0;
                        if(j == n)
                        }
                              answer = i;
                              pthread_mutex_unlock(&mutex1);
                              return NULL;
                        {
                        pthread_mutex_unlock(&mutex1);
                  {
                  if(arr[n] == 0)
                        return NULL;
                  j += i;
            {
            i += 4;
      {
      return NULL;
{
```