המכללה האקדמית בראודה המחלקה להנדסת תוכנה

14.3.2007

תכנות מתקדם 61617
מבחן סיום - מועד ב'
מרצה: ד"ר איתן רון

הערות

1) מותר כל חומר עזר.
2) זמן הבחינה - 3 שעות ללא אפשרות הארכה.
3) קרא בעיון את השאלות. אורכה של שאלה אינה בהכרח מעיד על הקושי שבה.
4) שאלות 1, 2, 3 הם שאלות תכנות.
התשובות לשאלות חייבות להיות בקוד C.

שאלה מספר 1 _(34 נקודות)

עליך לממש מערכת של של משאב ורוטינות התומכת בהעברת מחרוזות בין שתי תהליכים שיקיים את התנאים הבאים:

1. במשאב יש מקום למחרזת אחת בלבד, בגודל של עד 256 תוים.

2. מחרוזת נקראית פעם אחת בלבד, ברגע שהיא נקראית היא מתבטלת.

3. תהליך שרוצה לכתוב לשטח הזה מחרוזת, חייב להמתין (עם יש צורך בכך) עד שהשטח פנוי. השטח פנוי אם מעולם לא נכתב בו מחרוזת לקריאה, או שהמחרוזת האחרונה שנכתבה בו נקראה כבר.

4. תהליך שרוצה לקרוא מחרוזת מהמשאב חייב (אם יש צורך בכך) להמתין עד שתגיע הודעה שלא נקראה.

ניקוד מלא ינתן רק לפתרונות שיעמדו גם בתנאים הבאים:

5. (8 נקודות) תהליך ממתין לא ישתמש ב-busy wait, כלומר ההמתנה לא תתבצע על ידי בזבוז מכסות זמן.

6. (8 נקודות) על הרוטינות לדאוג לכך ש-
א. אם תהליך יתחיל לקרוא את המחרוזת, המחרוזת היא "שלו" ולא יקרא או יכתב ע"י התהליך האחר. אם תהליך אחד מתחיל לקרוא את המחרוזת והופסק באמצע, התהליך השני, אם הוא ירצה לקרוא או לכתוב לשטח, הוא יצטרך להמתין עד שהתהליך השני יקבל שוב שליטה ו יסיים.
ב. אם תהליך יתחיל לכתוב מחרוזת על השטח, השטח היא "שלו" ולא יקרא או יכתב ע"י התהליך האחר. אם תהליך אחד מתחיל לקרוא את המחרוזת והופסק באמצע, התהליך השני, אם הוא ירצה לקרוא או לכתוב לשטח, הוא יצטרך להמתין עד שהתהליך השני יקבל שוב שליטה ויסיים.

ההכרזות של הרוטינות שלך יהיו כדלקמן:

```
int atomic_buffer_init(int key);
```

המבקש את המשאב/ים הנחוצים ומאתחל אותם עם שם משני key.

```
int atomic_buffer_read(int shmid, char msg[]);
```

הקוראת את המחרוזת לפי החוקים שלעיל.

```
int atomic_buffer_write(int shmid, char msg[]);
```

ההכותבת את המחרוזת לפי החוקים שלעיל.

לדוגמא הפלט של התוכנית הבאה:

```
void main()
{
  int shmid;
  char msg1[256], msg2[256];
  int id;

  shmid = atomic_buffer_init(314159);

  id = fork();


  if (id == 0)
  {
    atomic_buffer_write(shmid, "Hello ");
    atomic_buffer_write(shmid, "World!");
    return;
  } /* if */

   atomic_buffer_read(shmid, msg1);
   atomic_buffer_read(shmid, msg2);

   printf("msg = %s %s\n",msg1, msg2);

} /* main */
```

יהיה:

```
% ./a.out
msg = Hello  World!
%
```

שאלה מספר 2 (33 נקודות)

בשאלה זו עליך לממש מנגנון של "לכידת חריגות" ב-C. ב-C חריגות באות לידי ביטוי אך ורק בסיגנלים, למשל SIGFPE מתרחש על כל תקלה אריטמטית כמו חלוקה באפס של מספר שלם או ממשי למשל.

עליך לממש פונקציה בשם my_try המקבלת כפרמטרים

1. מספר סיגנל שהתהליך יקבל אם יתרחש החריגה,
2. פוינטר לפונקציה שהיא תהיה קוד לביצוע,
3. פוינטר לפונקציה שתהיה קוד שיתבצע במקרה של התרחשות חריגה.

ניקוד מלא (7 נקודות הבדל) ינתן רק לפתרון שיחזיר את המצב לקדמותו (בכל הקשור ללכידת הסיגנלים) עם סיום ה-try ו(7 נקודות הבדל) גורמת לחידוש התוכנית ע"י בכל מקרה בצורה של חזרה מ-my_try.

ההכרזה של my_try יהיה:

```
void my_try(int  signum,
       void (*try_code)(void), void (*catch_code)(void));
```

לדוגמא, התוכנית הבאה מריצה פונקציה שמחלקת באפס, אבל אינה עפה משום שהפונקציה my_try לוכדת את הסיגנל SIGFPE ואם היא קוראת

```
void div_by_zero()
{
 int x, y;

 x = 0;
 y = 10/x;

} /* div_by_zero */

void catch_example()
{
   fprintf(stderr, "Arithmetic error\n");
} /* catch_example */

int main()
{

  my_try(SIGFPE, div_by_zero, catch_example);

  printf("Terminating regularly\n");

 return 0;

} /* main */
```

הפלט של התוכנית הזו תהיה:

```
% ./a.out
Arithmetic error
Terminating regularly

%
```

שאלה מספר 3 (33 נקודות)

עליך לכתוב תוכנית awk המסיע לבדיקת רמות של בלוקים של תוכניות C או שפות תכנות דומות אשר פותחים בלוקים ע"י התו "{" וסוגרים אותו ע"י התו "}". התוכנית מקבלת שם קובץ C כפרמטר לתוכנית ומדפיסה את תוכנו מחדש תוך ציון העומק של בתחילת כל שורה בסוגריים מרובעות.

לדוגמא, פלט של ריצה אפשרית של התוכנית יכולה להיות:

```
% awk -f depth.awk dec2hex.c
[0] /* dec2hex.c convert decimal to hexadecimal */
[0] #include <stdio.h>
[0]
```

```
[0] void main(int argc, char *argv[])
[1] {
[1]   long int x;
[1]
[1]   if(argc <2)
[2]   {
[2]     fprintf(stderr, "Usage: dec2hex decimal_number\n");
[2]     exit(0);
[2]   }
[1]
[1]    printf("argv[1] = %s\n", argv[1]);
[1]    sscanf(argv[1], "%ld", &x);
[1]    printf("%ld = 0x%lx\n", x, x);
[1]
[1]
[1] } /* main */
```

שאלה מספר 4 (בונוס 5 נקודות)

עליך לכתוב סקריפט של יוניקס הבודק אם בספריה הנוכחית ישנם זוגות של קבצים שתוכנם זהה ואם כן להדפיס הודעה על כל זוג כזה.

לדוגמא, פלט אפשרי של ריצה של הסקריפט שלך יכול להיות:

```
% test_idn
contents of file fibo.c is identical to file fibo1.c
contents of file fm.c is identical to file funm.c
%
```