24/8/2007

מבוא למערכות הפעלה – תרגיל 2

מרצה: ד"ר כרמי מרימוביץ.

בתרגיל זה יש לכתוב שתי תוכניות. את שתיהן יש לכתוב ב-Win32 וב-Posix.

תוכנית ראשונה:

1. עליכם לכתוב מערכת בה רצים מספר חוטים (threads) משני סוגים: H ו-O. על התוכנית הראשית לייצר מספר חוטים משני הסוגים בהתאם לארגומנטים שיתקבלו ממפעיל התוכנית.
2. חוט מחכה זמן אקראי קטן משנייה ("מייצר נתון"), ואז בהתאמה לסוג החוט קורא לפונקציה hinfo או oinfo.
3. כאשר לשני ספקים מסוג H ולספק אחד מסוג O יש נתון, על שגרת ה-info לקרוא לשגרה process ולתת לחוטים התקועים להמשיך לרוץ. חוט יתקע בשגרת ה-info אם אין מספיק נתונים לקרוא לשגרה process.
4. על החוטים להדפיס שיש נתון לפני הקריאה לפונקציית ה-info.
5. על השגרה process לספור כמה פעמים קראו לה, ולהדפיס זאת עם כל קריאה.

תוכנית שניה:

1. כיתבו תוכנית הפותרת את בעיית היצרן-צרכן עבור חוצץ(buffer=) חסום. בכיתה הוצג פתרון שמשתמש בשלושה סמפורים(semaphores=), כשאחד מהם שימש למניעה הדדית(mutex=). בתוכנית עליכם להשתמש בשני סמפורים ומונע הדדי(mutex=). היצרנים והצרכנים ירוצו (כמובן!) בחוטים(threads=) משלהם.
2. התוכנית תקבל שלושה ארגומנטים:
    - i. כמה זמן עליה "לישון" לפני סיום הריצה.
    - ii. מספר חוטי יצרן.
    - iii. מספר חוטי צרכן.
    - iv. מספר האיברים בחוצץ.
3. על התוכנית ליצור חוצץ, לאתחל אותו ואת אמצעי הסינכרון, ליצור חוטים במספר המבוקש ואז לישון, ועם ההתעוררות לסיים את הריצה.

4. תהליך היצרן ישן פרק זמן אקראי, מתעורר ומכניס מספר אקראי לחוצץ (וכן מדפיס זאת למסך) וחוזר חלילה.
5. תהליך הצרכן ישן פרק זמן אקראי, מתעורר ומוציא מספר מהחוצץ (ומדפיס את המספר שהוציא), וחוזר חלילה.

<u>פונקציות API שימושיות:</u>

1. Pthreads threads.

```
#include <pthread.h>
pthread_t tid;                  // Thread id
pthread_attr_t attr;            // Attributes

pthread_attr_init(&attr);       // Set default attributes
pthread_create(&tid, &attr, func, arg); // Start running at func, with arg

void *func(long param)          // The function the thread begins in.
```

2. Win32 threads.

```
#include <windows.h>
DWORD tid;
HANDLE thandle;

thandle = CreateThread(
    NULL,               // Default security attribute
    0,                  // Default stack size
    func,               // function to run
    arg,                // argument
    0,                  // default creation flag
    &tid                // thread id
);

DWORD WINAPI func(DWORD arg)        // thread function
```

3. Pthreads Mutex.

```
#include <pthread.h>
pthread_mutext_t mutex;

pthread_mutext_init(&mutext, NULL); //Create mutext
pthread_mutext_lock(&mutex);          // Acquire lock
// *** Critical section ***
pthread_mutext_unlock(&mutex);        // Relase lock
```

Pthreads Semaphore .4

```
#include <semaphore.h>
sem_t sem;

sem_init(&sem, 0, 5);     // Semaphore with initial value of 5.

sem_wait(&sem);           // Acquire
//  ** releavant code **
sem_post(&sem);           // Release
```

Win32 Mutex .5

```
#include <windows.h>
HANDLE mutex;

mutex = CreateMutex(NULL, FALSE, NULL);
WaitForSingleObject(mutex, INFINITE);
//  ** critical section **
ReleaseMutext(mutex);
```

Win32 Semaphores .6

```
#iinclude <windows.h>
HANDLE sem;

sem = CreateSemaphore(NULL, 0, 5, NULL);
WaitForSingleObject(sem, INFINITE);          // Acquire semaphore
```

```
//  ** releavant code **
ReleaseSemaphore(sem, 1, NULL);              // Release Semaphore
```

7. שימו לב, כדאי **מאוד (מאוד)** לקרוא את התיעוד של הפונקציות הנ"ל ולא להשתמש בהן באופן לא מושכל. זה **חלק מהתרגיל**.

אין לעבוד בקבוצות.

התיעוד משמעותי לציון.

שפות רלוונטיות: ++C/C **ללא** MFC/ATL וכיו"ב.

את התרגילים יש להגיש לתיבת התרגילים של הקורס.

מועד הגשה:6/9/2007.