**מערכות זמן אמת ותכנות מקבילי (157111)**

ד"ר ארי'ה טייטלבאום, מר יהונתן לויאן

**תרגיל 2 תשפ"א**

**שלב ראשון:**

שנה את התרגיל הראשון על מנת שיעבוד לפי האלגוריתם RMS. (עדיפות)

הרצה ובדיקה:

הרץ את אותה דוגמה כמו בתרגיל הראשון

**שלב שני:**

1. הוסף למערכת SMARTS מחלקה בשם Mutex. מחלקה זו תפעיל מנגנון סנכרון של סמפור בינארי מסוג Mutex (האלגוריתם לסמפור מתואר במצגת "פרק 4".)

ממש בעזרת תור עדיפויות (אין צורך לממש FIFO)

יש לדאוג שהמתודות Acquire ו Release תהיינה אטומיות.

1. יש לשנות את שיטת SMARTS בטיפול בקטע אטומי, כך ש:

* הדגל לא יהיה גלובלי אלא לוקלי, כלומר לכל תהליך נצמד דגל.
* הדגל חוסם רק פסיקות שעון שמקורם בחומרה.

הרצה ובדיקה:

שנה את אפליקציית הדוגמא והפוך כל קטע אטומי לקטע קריטי.

**שלב שלישי:**

הוסף לסמפור אפשרות של Priority Inheritance.

הרצה ובדיקה:

התהליכים בעלי העדיפות הגבוהה ימתינו ל event בתחילת הרצתם.

התהליכים בעלי העדיפות הנמוכה ישחררו את התהליך בעל העדיפות שמעליו, לאחר תפיסת המשאב (גם הגבוה ינסה לתפוס משאב לכשיתעורר).

כתוב אפליקציה שתמחיש את בעיית Unbounded Priority Inversion של קטעים קריטיים:

1. דוגמא בסיסית להרצה (ללא הירושה).

הרץ את האפליקציה ורשום זמני המתנה ממוצע של התהליכים למשאבים השונים.

הרץ פעמיים, עם ובלי הורשה, והשווה את ממוצעי ההמתנה.

1. הרץ דוגמא זאת :

השתמש בשני משאבים וארבעה תהליכים. הראשון ישתף משאב עם השני, והשלישי עם הרביעי.

הוסף שני תהליכים בעלי עדיפות ביניים, אחד בין הראשון לשני, ואחד נוסף בין השלישי לרביעי. תהליכים אלו אינם משתפים משאבים, הם עשויים להפוך את הבעיה ל unbounded.

הערה:

בהרצה אין לביים קינון של קטעים קריטיים.

**עד כאן ניתן לקבל ציון מקסימאלי 90.**

**שלב רביעי (בונוס , שניתן לקבל ציון 100 )**

עליך לדאוג שההורשה תהיה רקורסיבית (בונוס).

כתוב אפליקציה שתמחיש את בעיית Unbounded Priority Inversion בתוספת קינון של קטעים קריטיים:

* הרץ 5 תהליכים בעלי עדיפויות שונות, המשתפים בזוגות 4 משאבים, כלומר: התהליך הגבוה וזה שאחריו ישתפו משאב אחד, השני בעדיפות ישתף משאב עם השלישי, השלישי עם הרביעי והרביעי עם החמישי.

כך יוצא שהתהליך השני, השלישי והרביעי משתמשים כל אחד ב 2 משאבים.

* 4 התהליכים בעלי העדיפות הגבוהה ימתינו ל event בתחילת הרצתם.

התהליכים בעלי העדיפות הנמוכה ישחררו את התהליך בעל העדיפות שמעליו, לאחר תפיסת המשאב (גם הגבוה ינסה לתפוס משאב לכשיתעורר).

בהמשך התהליך השני, השלישי והרביעי יבקשו את המשאב השני.

באופן זה ייווצר קינון של קטעים קריטיים, כלומר:

התהליך הגבוה רדום בהמתנה למשאב שתפוס ע"י השני, השני רדום בגלל המשאב שתפוס ע"י השלישי וכן הלאה.

הוסף עוד 3 תהליכים שאינם משתפים משאבים, הם עשויים להפוך את הבעיה ל unbounded.

תהליכים אלו הם בעלי עדיפות הביניים, לא יניחו לנמוכים לרוץ ולשחרר את המשאבים לגבוהים.

הרצות ובדיקות:

הרץ את האפליקציה ורשום זמני המתנה ממוצע של התהליכים למשאבים השונים.

הרץ פעמיים, עם ובלי הורשה, והשווה את ממוצעי ההמתנה.

טיפים:

מאחר ואין אנו יכולים לראות פלט ארוך

יש להשתמש בכתיבה לקובץ (בבדיקות ובהגשת התרגיל)

FILE\* myOutput=fopen("test.txt",w);

printf(myOutput,"A");

מאחר והתסריטים ארוכים

כדאי להגדיל את זמן המחזור / עדיפות

SMARTS.declareTask(a,'A', 400,X);

SMARTS.declareTask(b,'B',401,X);

SMARTS.declareTask(c,'C',402,X);

מספר הפעמים שהם עובדים שווה ((X,

העדיפות של a היא הגבוהה מבניהם וזמן מחזור גדול וסמוך בכדי שלא ייווצר מצב שאחד 'עולה' ומחכה ל אירוע ואין מי שישלח לו.

שליחה וקבלת אירוע.

שליחה:

evX.sendEvent('A',NULL,0)

לאיזו משימה שולחים את האירוע (האירוע יהיה זמין רק לו).

תוכן האירוע.

אירוע סינכרוני (1) או א-סנכרוני (0).

קבלה:

char ch;

y=evX.waitEvent(ch);

בקבלת אירוע מקבלים את הערך שנשלח ( left value אופצונאלי).

ומי השולח. (ch)

**מסירת התרגיל עד השבוע האחרון של הסמסטר.**

**בהצלחה !!!!**