

עבודה 3 מערכות הפעלה:

שאלה 1:

א. נדבר על שני סוגי זיכרון שזמינים עבור פרוסס וטארד – heap ו stack, לכל טראד יש את המקום שלו ב stack אבל כל הטראדים המשותפים לאותו פרוסס יחלקו את ה heap, בגלל זה טראדים נקראים "קלים יותר" בגלל שיש לכל אחד מהם את ה stack שלו אבל יכולים לגשת למידע משותף.

בגלל שטראדים שבתוך הפרוסס חולקים את אותו מרחב כתובות כמו הפרוסס, הגישה בין הטראדים קלה יותר שזה יתרון. (מצד שני חיסרון בכך יכול להיות שכאשר יש בעיה בטראד אחד בתוך הפרוסס דבר זה יכול להשפיע על שאר הטראדים הנמצאים תחת אותו פרוסס).

ב. חסם עליון למהירות התוכנית כתלות במספר המעבדים:

$$SPEEDUP \leq \frac{1}{S + \frac{1-S}{N}}$$

S - חלק סיראלי של התוכנית

N - כמות המעבדים

SPEEDUP - מהירות מקסימאלית של תוכנית

עבור 30% ו 70% COREI 2:

$$SPEEDUP \leq \frac{1}{0.3 + \frac{1-0.3}{2}} = \frac{20}{13}$$

עבור 30% ו 70% COREI 4:

$$SPEEDUP \leq \frac{1}{0.3 + \frac{1-0.3}{4}} = \frac{40}{19}$$

עבור 30% ו 70% COREI 8:

$$SPEEDUP \leq \frac{1}{0.3 + \frac{1-0.3}{8}} = \frac{80}{31}$$

עבור 30% ו-70% CORE: 64:

$$SPEEDUP \leq \frac{1}{0.3 + \frac{1-0.3}{64}} = \frac{640}{199}$$

עבור 8% ו-92% CORE: 2:

$$SPEEDUP \leq \frac{1}{0.08 + \frac{1-0.08}{2}} = \frac{50}{27}$$

עבור 8% ו-92% CORE: 4:

$$SPEEDUP \leq \frac{1}{0.08 + \frac{1-0.08}{4}} = \frac{100}{31}$$

עבור 8% ו-92% CORE: 8:

$$SPEEDUP \leq \frac{1}{0.08 + \frac{1-0.08}{8}} = \frac{200}{39}$$

עבור 8% ו-92% CORE: 64:

$$SPEEDUP \leq \frac{1}{0.08 + \frac{1-0.08}{64}} = \frac{1600}{151}$$

ג. השימוש בטרמד פול הופך תוכניות ליותר יעילות מכיוון שהוא :

save initiation time , less overhead

לדוגמה תוכנית שרצה יותר טוב עם בריכת טראדים:

יטויב- מקבל פניות ממשתמשים כל הזמן בכל המקרים זה אותה עבודה

של שליחת סרטון מכיוון שזו אותה הפונקציה לא צריך כל הזמן ליצור

טרמד חדש.

שאלה 3:

.א

```
91
92 1 reference
class mySemaphore
93 {
94     int size;
95     static int count;
96     Mutex[] mutexs;
97     Thread[] myThreads;
98
99 0 references
public mySemaphore(int theSize)
100 {
101     size = theSize;
102     count = 0;
103     mutexs = new Mutex[size];
104     myThreads = new Thread[size];
105     for (int i = 0; i < size; i++)
106     {
107         mutexs[i] = new Mutex();
108     }
109
110 }
111 0 references
public void wait()
112 {
113     while (count >= size) { }
114     int i = 0;
115     for (i = 0; i < mutexs.Length; i++)
116     {
117         mutexs[i].WaitOne(1);
118     }
119     myThreads[i] = Thread.CurrentThread;
120     count++;
121 }
122
123 0 references
public void Release()
124 {
125     for(int i = 0; i < size; i++)
126     {
127         if(myThreads[i] == Thread.CurrentThread)
128         {
129             mutexs[i].ReleaseMutex();
130             count--;
131             break;
132         }
133     }
134 }
135
136 }
```

ב. נשמור כמו במין עץ בינארי עבור כל node:

b[level,2node+1], b[level,2node], turn[level,node]

עבור כל פרוסס:

Level ,node ,id

Program for process i:

Node:=i

For level 0 to log n-1 do:

Id=node mod 2;

Node= [node/2]

b[level,2node+id]

turn[level ,node] = id

await (b[level,2node+1-id] = false or turn[level ,node]=1-id)

end do

CS

For level = log n-1 down_to 0 do:

Node = [i / 2^level]

b[level,2node+id]

end do

ג. נרשום פסאדו קוד עבור כל אחד מה statements :

1. אתחול ארבעה Semaphore עם ערך 0 ונקרא להם S1,S2,S3,S4

statement S1:

#code Statement 1

Signal(S1);

statement S2:

Wait(S1);

#code Statement 2

Signal(S2);

statement S3:

Wait(S2);

#code Statement 3

Signal(S3);

statement S4:

Wait(S3);

#code Statement 4

Signal(S4);

statement S5:

Wait(S4);

#code Statement 5

ד. deadlock מתאר סיטואציה שבה שני טראדים או יותר נחסמים לנצח וזאת מכיוון שהם מחכים אחד לשני, deadlock מופיע כאשר מספר טראדים צריכים את אותו מנעולים אך משיגים אותם בסדר שונה. תוכנית עם כמות טראדים גדולה עלולה לסבול ממצב של deadlock שמילת המפתח המסונכרנת גורמת לחסימת הטרמד המבוצע בזמן ההמתנה למנעול או למוניטור המשויכים לאובייקט שצוין.

דוגמה לקוד עם deadlock:

1. Semaphore s1 = new semaphore(0,1)

2. Semaphore s2 = new semaphore(0,1)

3. Two processes P1 and P2;

P1:

4. Wait(s1)

5. Wait(s2)

P2:

6. Wait(s2)

7. Wait(s1)

הסבר: P1 מבצע את שורה ארבע ומיד לאחר מכן יש P2 context switch מבצע את שורה שש וקעת ישנו מצב של דד-לוק מכיוון שP1 מחכה למנעול שP2 תפס ואילו P2 מחכה למנעול שP1 תפס ואף פרוסס לא משחרר את המנעול שלו.

ה. השוואה בין שני האלגוריתמים:

<u>Petersonss</u>	<u>Dekkers</u>	נקודות להשוואה
מינימום 2 הכי הרבה n , דבר המקנה לו ייתרון.	אי אפשר להרחיב אותו ליותר משני טראדים במקסימום ולכן במערכות גדולות דבר שיכול לגרום לאי נוחות.	כמות פרוססים מקסימאלית שעליהם ניתן לבצע את האלגוריתם
לא ייתכן מצב כזה	ייתכן מצב כזה	מצב שבו פרוסס אחד מקבל יותר זמן באזור הקריטי
משתמש רק בזיכרון משותף לסינכרון רק פעולות הLODE והSTORE צריכות להיות אטומיות	כל טארד יכול לצאת לפועל רק בסנכרון מאוד קפדני	

שאלה 4:

בסעיף זה השתמשנו בסוג מנעול ששמו ReaderWriterLockSlim והוא עוזר לנו ליעל את ריצת התוכנית שלנו, מכיוון שהוא מאפשר להרבה קוראים לקרוא את אותו מסמך יחד ובעצם מפחית את העומס על אותו מנעול (עדין כל כותב צריך לקבל תור בנפרד).

החלטנו שעבור סכום מנעולים מינמאלי נשתמש במנעול עבור כל פונקציה וכאשר המשתמש יקרא לפונקציה הוא בעצם ינעל אותה ויוכל לעשות בה שימוש ללא חשש מכניסת טראדים אחרים השתמשנו באסטרטגיה זאת מכיוון שרצינו להשקיע את רוב הזמן על שאלה 5 ולעשות אותה בצורה המיטבית ולכן בחרנו לנעול את כל הפונקציות בשאלה זו.

שאלה 5:

א. הסבר:

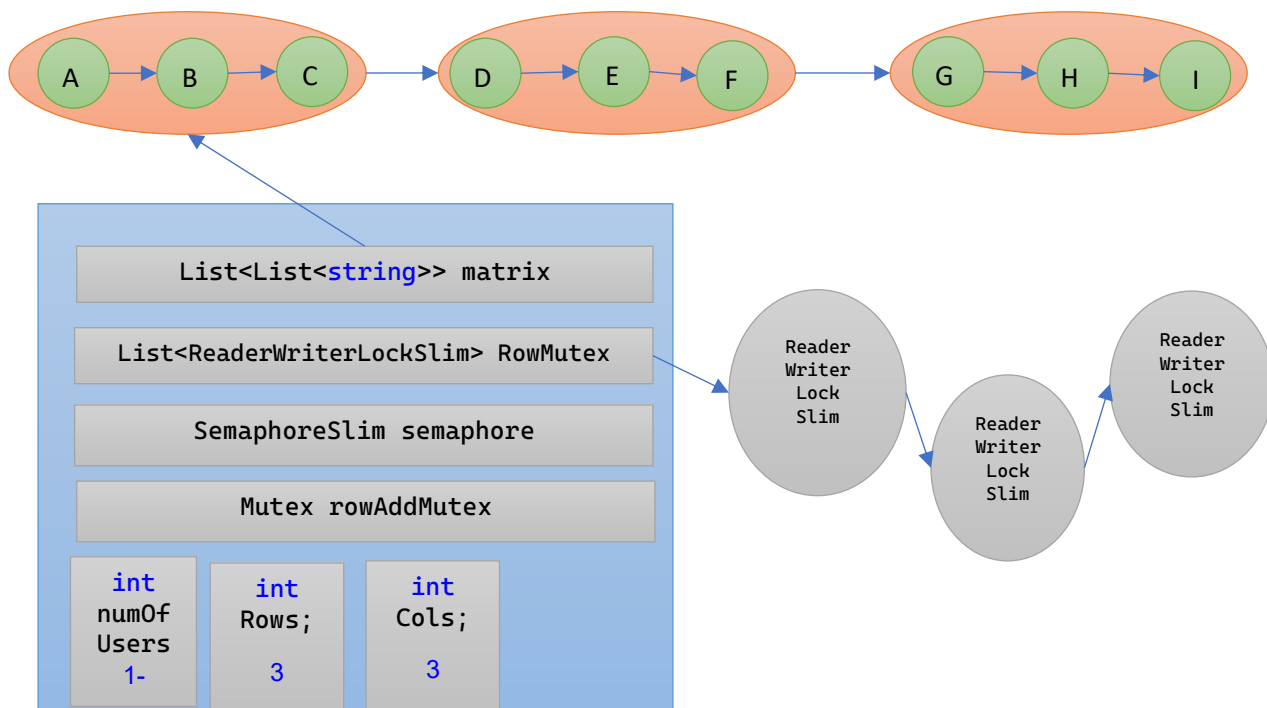
האובייקט שלנו מחולק לפי שורות כאשר אנו רוצים לבצע פעולות על האובייקט שלנו אנו תמיד נועלים אך ורק 2 שורות לכל היותר בכל הפונקציות (דבר זה עוזר לתוכנית שלנו לרוץ בצורה יעילה יותר מכיוון שאיננו נועלים את כל הפונקציה או את כל השורות וגם לא נועל גם עמודות וגם שורות - כל פעם נועל רק 2 שורות בכל התוכנית). בנוסף אנו משתמשים בסוג מנעול ששמו ReaderWriterLockSlim והוא עוזר לנו ליעל את ריצת התוכנית שלנו, מכיוון שהוא מאפשר להרבה קוראים לקרוא את אותו מסמך יחד ובעצם מפחית את העומס על אותו מנעול (עדין כל כותב צריך לקבל תור בנפרד).

הכנסנו סמופור שאחראי על כל האובייקט מי נכנס אליו ומי יוצא ומתוחזק בתוך כל אחת מהפונקציות, הכנסנו עוד מנעול על חלק קטן מפונקציית ההוספת שורה כי רצינו להפוך חתיכת קוד מאוד קטנה לאטומית (נציין שהתוכנית שלנו לא נתקעת על זה מכיוון שחלק קוד זה קטן).

אנחנו משתמשים בכמות מפתחות ככמות השורות שלנו בתוכנית והיא מיוצגת לפי רשימה שאנחנו מחזיקים באובייקט שלנו.

למשל עבור הSpreadsheetApp הבא :

A	B	C
D	E	F
G	H	I



סעיף ג':

Form1

1	2	3	4	5	6	7	8	9	10
test cell00	test cell01	test cell02	test cell03	test cell04	test cell05	test cell06	test cell07	test cell08	test cell09
test cell10	test cell11	test cell12	test cell13	test cell14	test cell15	test cell16	test cell17	test cell18	test cell19
test cell20	test cell21	test cell22	test cell23	test cell24	test cell25	test cell26	test cell27	test cell28	test cell29
test cell30	test cell31	test cell32	test cell33	test cell34	test cell35	test cell36	test cell37	test cell38	test cell39
test cell40	test cell41	test cell42	test cell43	test cell44	test cell45	test cell46	test cell47	test cell48	test cell49
test cell50	test cell51	test cell52	test cell53	test cell54	test cell55	test cell56	test cell57	test cell58	test cell59
test cell60	test cell61	test cell62	test cell63	test cell64	test cell65	test cell66	test cell67	test cell68	test cell69
test cell70	test cell71	test cell72	test cell73	test cell74	test cell75	test cell76	test cell77	test cell78	test cell79
test cell80	test cell81	test cell82	test cell83	test cell84	test cell85	test cell86	test cell87	test cell88	test cell89
test cell90	test cell91	test cell92	test cell93	test cell94	test cell95	test cell96	test cell97	test cell98	test cell99

find all Get Cell Load Spreadsheet Save Spreadsheet

insert string you want to find insert string you want to find File Name

Bool Sensitive(true/false)

Clear Color

למשל הפעלת הפונקציה Get Cell אשר צובעת את החיפוש:

Form1

1	2	3	4	5	6	7	8	9	10
test cell00	test cell01	test cell02	test cell03	test cell04	test cell05	test cell06	test cell07	test cell08	test cell09
test cell10	test cell11	test cell12	test cell13	test cell14	test cell15	test cell16	test cell17	test cell18	test cell19
test cell20	test cell21	test cell22	test cell23	test cell24	test cell25	test cell26	test cell27	test cell28	test cell29
test cell30	test cell31	test cell32	test cell33	test cell34	test cell35	test cell36	test cell37	test cell38	test cell39
test cell40	test cell41	test cell42	test cell43	test cell44	test cell45	test cell46	test cell47	test cell48	test cell49
test cell50	test cell51	test cell52	test cell53	test cell54	test cell55	test cell56	test cell57	test cell58	test cell59
test cell60	test cell61	test cell62	test cell63	test cell64	test cell65	test cell66	test cell67	test cell68	test cell69
test cell70	test cell71	test cell72	test cell73	test cell74	test cell75	test cell76	test cell77	test cell78	test cell79
test cell80	test cell81	test cell82	test cell83	test cell84	test cell85	test cell86	test cell87	test cell88	test cell89
test cell90	test cell91	test cell92	test cell93	test cell94	test cell95	test cell96	test cell97	test cell98	test cell99

find all Get Cell Load Spreadsheet Save Spreadsheet

insert string you want to find insert string you want to find File Name

Bool Sensitive(true/false)

Clear Color