תכנות מרובה ליבות - תרגול 2#

הגדרות:

- אינטרוול: שני אירועים – כניסה ויציאה ומה שביניהם.
- Mutual Exclusion: בין שני קטעים קריטיים יש ME אם הם לא חופפים (אחד לפני השני או הפוך) – הגדרת שני אירועים שיש ביניהם סדר.
- Lock free: המערכת מתקדמת, כלומר תמיד יש מישהו שמתקדם והמערכת לא עוצרת. מבחינת מנעול – לא קורה שאף אחד לא מצליח לתפוס את המנעול.
- Starvation free: כל משתמש יתפוס את המנעול מתישהו, ללא תלות בין החוטים.

בשתי ההגדרות האחרונות הכוונה היא שאלגוריתם מימוש המנעול יהיה כזה, יתכן שהאלגוריתם המשתמש במנעול לא יענה על הגדרות אלו.

פתרון תרגיל בית 1, שאלת Flanky lock:

<u>הערה</u>: לכל thread יש מזהה יחודי שניתן להשיגו ע"י Thread.currentThread().getId().

כדי ליצור מספור רציף ל-thread (הנ"ל לא מבטיח מספור נוח) הוא שימוש ב-ThreadLocal<Integer> id ודורסים לו את מתודת initialValue() המייצרת מספר unique לחוט ושומרת אותו במשתנה לוקאלי של החוט. בקריאה הבאה של המתודה יוחזר המספר שניתן לו בהתחלה. דוגמא לכך תופיע באתר. באמצעות ThreadLocal ניתן ליצור משתנים פר חוט.

<u>הוכחה שה-Flanky הוא מנעול</u>:

נניח בשלילה שהוא לא מנעול, כלומר יש לפחות שני חוטים שנכנסו ל-CS, ומכאן שניים ראו את $busy = false$, ואז שניים עברו את $turn = me$, אבל אז אחד מהם לא ראה $turn! = me$ ולכן לא נכנס ל-CS.

<u>הוכחה שהמנעול לא starvation free</u>: פשוט.

<u>הוכחה שהמנעול לא dead-lock free</u>: גם פשוט.

שאלת Bouncer (שקופית 11):

<u>הוכחה שלכל היותר אחד עוצר</u>:

נניח בשלילה ששניים עצרו, ולכן שניהם ראו $last == i$, ואז בודאות אחד כתב את עצמו, והגיע ל-STOP. אז לא יתכן שהשני הגיע לשם, כי אז כבר $goRight = true$ ולכן הוא הלך ימינה ולא עצר. תיאור מהלך שני ה-threads בשקופית 13-14.

אלגוריתם Bakery:

```
class Bakery implements Lock {
  …
 public void lock() {
  flag[i]  = true;
  label[i] = max(label[0], …,label[n-1])+1;
  while (∃k flag[k]
           && (label[i],i) > (label[k],k));
 }
```

מימוש זה תיאורטי בלבד שכן הוא לא יעיל.

הוכחה שיש צורך ב-N רגיסטרים MRSW:

תחילה, כל חוט כותב לאיזשהו רגיסטר, ולכן לא יתכן שחוט אחד לא יכתוב כלל לרגיסטר, כי הוא חייב להודיע שהוא נכנס למנעול איכשהו. מכאן, כל חוט שרוצה להיכנס למנעול צריך לכתוב לרגיסטר.

<u>בסיס האינדוקציה</u>: לא יכול להיות שעברו שני חוטים מספיק רגיסטר אחד למימוש מנעול: אם שני חוטים מגיעים לרגיסטר המשותף, מישהו מהם יכתוב על הרגיסטר. נניח ש-A רץ עד רגע לפני כתיבה לרגיסטר. כאן ניתן ל-B לתפוס את המנעול ונכנס לקטע הקריטי. כעת, A ממשיך ודורס את הכתיבה של B לרגיסטר, ומכאן ש-A לא יודע ש-B תפס את המנעול.

<u>הגדלת האינדוקציה</u>: נניח שעבור 3 חוטים יש 2 רגיסטרים. נגיע למצב ש-B רגע לפני כתיבה לרגיסטר שלו, חוט C יגיע גם רגע לפני כתיבה לרגיסטר שלו, A נכנס לקטע הקריטי (בדרך כותב ללא הגבלת הכלליות לרגיסטר של B). אז B (במקרה זה) יכנס לקטע הקריטי (C לא יכנס, אבל A ו-B שניהם יחד בקטע הקריטי).

מדוע ניתן לאמר ששני חוטים B ו-C מגיעים לכיסוי, כלומר שניהם רגע לפני כתיבה לרגיסטרים? לפי עקרון שובך היונים, בשלושה סיבובים חוט מסויים יכסה רגיסטר כלשהו פעמיים לפחות. נניח B רץ עד רגע לפני שכותב לרגיסטר שלו, A רץ גם עוצר רגע לפני הכתיבה לרגיסטר שלו (אם כתב לרגיסטר של B, כאשר B ימשיך יהיו שניים בקטע הקריטי וסיימנו). כרגע אין עדות באף אחד מהרגיסטרים שמישהו מ-A ו-B מנסה לתפוס את המנעול. כעת אם יבוא C הוא חייב להצליח לתפוס את המנעול. יכול להיות ש-A הספיק לכתוב לרגיסטר של B. בשלב הבא, B יכתוב לרגיסטר שלו ואולי ידרוס מה ש-A כתב.