BANK (RELEASE NOTES)

מגישים: כרמל רבינוביץ' 302958145 , אמיר ליבנה 201301405

להלן הסבר על חלקי הקוד ואופן המימוש בו בחרנו במטרה לבנות את הבנק, כפי שפורט במסמך הדרישות באתר. כמו-כן, מפורטות הנחות שביצענו, במקרים בהן ההוראות השאירו מקום לספק.

כמו-כן, פירוט נרחב בנוגע למימוש הפרטני של הפונקציות השונות מופיע לאורך הקוד כמקובל.

1. מנגנון קוראים-כותבים:

a. בחרנו לממש מנגנון קוראים-כותבים בצורה הבאה: מנעול "כתיבה", אשר מונע הן כתיבה והן קריאה מהקטע הקריטי, ובנוסף לכך מונה הסופר את כמות הקוראים בזמן נתון, אשר מוגן במנעול "קריאה". בכל פעם שקורא "נכנס" לקטע הקריטי תופעל הפונקציה readerEnter(), אשר תעלה את המונה ב-1 בצורה בטוחה (ובמידה ומדובר בקורא הראשון, תנעל גם את מנעול ה"כתיבה"), ובכל פעם שייצא – תופעל הפונקציה readerLeave(), אשר תוריד את המונה ב-1 (ובמידה ומדובר בקורא האחרון, תשחרר את מנעול ה"כתיבה"). בכך אנו מאפשרים למספר threads לבצע פעולת קריאה באופן סימולטני, אך לא יתכן כי תבוצע פעולת קריאה ופעולת כתיבה, או שתי פעולות כתיבה, בו-זמנית.

2. מבני הנתונים ומנעולים:

a. בחרנו לממש מחלקה של חשבון ומחלקה של בנק, ובנוסף struct לכל ATM.

b. מחלקת bank:

i. לכל בנק (בתוכנית זו קיים רק בנק אחד – best_bank, אך ניתן להרחיב למספר בנקים בקלות) מפה המחזיקה את כל החשבונות שלו, וכן משתנה המכיל את יתרת הבנק.

ii. המפה מוגנת באמצעות מנגנון קוראים-כותבים (כפי שפורט לעיל). פעולת כתיבה תיחשב כהכנסה או הוצאה של חשבון ל-DB (כלומר פעולות open ו-quit), וכן הדפסת מצב הבנק (פירוט תחת סעיף 3.b). בכל שאר הפעולות, המצריכות גישה לחשבון ספציפי, יופעל מנעול הקריאה (באופן אינטואיטיבי – מבוצעת קריאה מה-DB על מנת למצוא חשבון מסוים, ולאחר שמצאנו את החשבון נוכל לבצע עליו פעולות פנימיות).

iii. יתרת הבנק מוגנת גם היא מפני עדכון וקריאה בו-זמנית, בעזרת מנעול נוסף. במקרה זה ישנו רק קורא אחד אפשרי (ה-thread אשר אחראי על הדפסת סטטוס הבנק, בפי שיפורט בהמשך), ולכן לא היה צורך לממש מנגנון קוראים-כותבים מלא, אלא מנעול יחיד בלבד.

iv. בנוסף, לבנק גישה למנעולים המגינים על כל חשבון וחשבון (וממומשים גם הם בצורת קוראים-כותבים, כפי שיפורט בסעיף הבא), ובעת ביצוע פעולה על חשבון מסויים, הבנק ינעל את החשבון הספציפי לקריאה/כתיבה בהתאם לצורך (לדוגמא עבור פעולת העברה הבנק ינעל לכתיבה את שני החשבונות). בכך אנו מאפשרים מקביליות לביצוע פעולות על מספר חשבונות במקביל.

v. כמו-כן, נציין כי הבנק אחראי על כלל ההדפסות לקובץ הלוג. ההדפסות מוגנות מפני התנגשויות בעזרת מנעול, כפי שיפורט בסעיף 3.d.

c. מחלקת account:

i. כל account מחזיק את משתני החשבון (ID, סיסמא ויתרת חשבון).

ii. לכל חשבון מנעול כתיבה ומנעול קריאה, בהתאם למימוש מנגנון קוראים-כותבים שפורט לעיל. הבנק אחראי להפעלת המנעולים הנ"ל.

iii. פעולות כתיבה בחשבון – עדכון יתרה, גביית עמלה.

iv. פעולות קריאה בחשבון – בירור יתרה, בירור סיסמא, בירור מספר חשבון.

d. כל כספומט ממומש ע"י struct המחזיק את מספר הכספומט ואת השם של קובץ ה-input של אותו כספומט, המכיל את הפעולות של אותו כספומט. מבנים אלה מועברים ל-thread-ים של כל ATM, על מנת שתהיה אוריינטציה ברורה בין thread ל-ATM המתאים לו (כפי שיפורט בסעיף הבא).

3. מקביליות:

a. בחרנו לממש כל ATM ב-thread נפרד (כלומר נקבל N thread-ים), כך שכל הכספומטים פועלים במקביל. כל ATM קורא את קובץ הפקודות המתאים אליו בצורה סיריאלית ובכך שומר על סדר הפקודות, כאשר התנגשויות בין פעולות של כספומטים שונים מנוהלות ע"י המנעולים שפורטו בסעיפים 1 ו-2. נציין כי בהתאם לתשובות בפורום השאלות במודל, יצאנו מנקודת הנחה כי הפקודות עצמן בפורמט חוקי, מבחינת כמות ארגומנטים, סדר וצורה.

b. Thread נוסף אחראי על הדפסת מצב הבנק.

i. בהתאם לתשובות בפורום השאלות במודל, ההדפסה הינה פעולה אשר אמורה לקחת snapshot של ה-DB, ולכן לפני כל הדפסה יופעל מנעול ה"קריאה" של המפה של חשבונות הבנק. בכך, בזמן פעולת ההדפסה לא יכולות להתבצע פעולות כתיבה או קריאה באף חשבון בבנק.

ii. ההדפסה נעשית למסך (ולא לקובץ ה-log), ומכיוון שזהו ה-thread היחיד שמדפיס למסך, אין צורך בהגנה נוספת במקרה זה.

c. Thread נוסף אחראי על גביית העמלה מכלל החשבונות:

i. בכך איטרציה של גביית עמלה, מחושב באופן אקראי אחוז העמלה (בין 2-4%), אשר זהה לכלל החשבונות בבנק. בכל איטרציה אחוז זה יוגרל מחדש.

ii. גביית העמלה היא פעולת קריאה ביחס למבנה הנתונים של הבנק (שכן אנו ניגשים לכל חשבון בבנק), ופעולת כתיבה ביחס לכל חשבון (שכן אנו מעדכנים את כמות הכסף בכל חשבון).

iii. בכך, יצאנו מנקודת הנחה שבעת גביית העמלה מהחשבון הראשון לדוגמא, יכולות להתבצע פעולות על חשבונות אחרים, אשר בפועל ישפיעו על ערך גביית העמלה הקרוב מאותם חשבונות. יש לציין כי אחוז העמלה הנגבה עדיין יהיה זהה לכלל החשבונות באותה איטרציה.

d. כדי להגן על הכתיבה לקובץ ה-log, קיים מנעול כתיבה לקובץ הלוג, כך שבכל רגע נתון רק כספומט אחד / גביית העמלה יכולים לכתוב לקובץ. הדבר פוגע במקביליות אולם זוהי דרישה בתרגיל.

e. נציין כי כלל המנעולים בתוכנית ממומשים ע"י semaphors. הסיבה לכך היא שלאחר מחקר באינטרנט, מצאנו כי ישנה בעיה לאתחל מנעולים מסוג pthread_mutex ב-thread שונה מה-thread בו מבצעים את פעולת ה-join. מבחינת אופן המימוש הדבר מתבטא בהבדלי syntax בלבד, וחסר משמעות מבחינה לוגית.

4. Flow של התכנית:

a. התוכנית יוצרת קובץ log.txt ומאתחלת את המנעולים הרלוונטיים.

b. התכנית מאתחלת את N ה-struct-ים של ה-ATM.

c. התכניות יוצרת ומאתחלת את thread גביית העמלה, לאחר מכן את thread הדפסת מצב הבנק, ולבסוף thread לכל ATM. ה-thread-ים של גביית העמלה והדפסת הבנק נעצרים בעזרת flag שמועברים אליהם כפרמטר. נציין כי דגלים אלה מוגנים גם הם במנעולים, על מנת למנוע קריאה ועדכון בו-זמנית.

d. כעת נבצע join לכל ה-threadים.

e. כאשר כל ה-thread-ים של כל ה-ATM סיימו את פעולתם ומתים, משנים את הערך של הדגל `ATMs_active_flag` ל-FALSE, וכך נעצר thread גביית העמלה.

f. כאשר ה-thread של של גביית העמלה סיים את פעולתו ומת, משנים את הערך של הדגל `prog_running_flag` ל-FALSE ובכך נעצר thread הדפסת הבנק.