ב"ה

תרגיל מס' 4– Threads, Mutex, Semaphores, and Shared Memory

הוראות הגשה

- שאלות בנוגע לתרגיל נא להפנות לצוות הקורס בכתובת os.89231@gmail.com (לא יהיה מענה למיילים שיגיעו לכתובות מייל אחרות):
- מענה יינתן אך ורק בין התאריכים: 15-22/06/2017 (כולל)

אנא היערכו בהתאם !!

- מועד אחרון להגשה: 30/06/17 23:59.
- יש לשלוח את הקבצים באמצעות האתר:
http://help.cs.biu.ac.il/submit.htm
לפני חלוף התאריך הנקוב לעיל.
- שם ההגשה של תרגיל 4 : ex4
- יש להקפיד מאוד על כל הוראות עיצוב הקלט והפלט כפי שלמדתם בקורס C (בדיוק אותו coding style)
- **עליכם לבדוק בנוגע לכל פונקציה האם היא הצליחה או לא**, אם היא לא הצליחה יש לתת הודעה מתאימה ל STDERR ולסיים את התכנית (בצורה נקייה כמובן).
- אין להשתמש בפונקציה ספריה אם ראינו בכיתה SYSTEMCALL מקבילה לה.
- **להזכירכם, העבודה היא אישית. "עבודה משותפת" דינה כהעתקה.**
- אין להדפיס שום דבר מעבר למה שנתבקש בתרגיל.
- יש לוודא שהתרגיל מתקמפל ורץ על ה U2 ללא שגיאות/אזהרות.

Threads, Mutex, Semaphores, and Shared Memory

הוראות כלליות לתרגיל

בתרגיל זה אתם צריכים לממש קוד שרת המריץ בתוכו Thread Pool וקוד לקוח אשר שולח לשרת בקשות.

כאשר השרת מקבל בקשה הוא צריך להוסיף את הבקשה לתור "עבודות" (jobs queue).

אחד מה threads אשר נמצאים ב thread pool צריך לקחת את הבקשה ולבצע אותה ברגע שהבקשה נכנסת לתור העבודות.

בתרגיל זה אין להשתמש ב SLEEP, ALARM או SIGNALS ככלי סנכרון או בכלל (מותר להשתמש רק במקומות שהוגדרו במפורש בתרגיל).

בנוסף אין להשתמש במנגנוני IPC מלבד המקומות בהם כתוב במפורש בתרגיל שאתם צריכים להשתמש בהם.

לקבלת מידע בנושא Thread Pool ניתן להיכנס לקישור הבא:
http://en.wikipedia.org/wiki/Thread_pool_pattern

הבודק ייצר מופע אחד מסוג שרת והרבה מופעים מסוג לקוח, עליכם להבין כיצד לסנכרן בין התהליכים וה threads וכיצד להימנע מ busy waiting.

<u>הנחיות עבור ex41 – צד לקוח</u>

- שם התרגיל: ex4
- שם קבצי מקור (source file) שיש לשלוח: ex41.c

על הלקוח להתחבר לזיכרון משותף אשר השרת יצר בעזרת מפתח ייחודי (שם הקובץ למפתח יהיה *<yourID>*.txt כלומר תעודת הזהות שלכם עם סיומת txt. , התו שתשתמשו בו נתון להחלטתכם, הוא צריך להיות זהה לזה שהשרת השתמש בו ביצירת הזיכרון המשותף). לדוגמא – 123456789.txt, 'A'

לאחר ההתחברות על התכנית שלכם:

1. להציג למשתמש את הבקשה הבאה:

Please enter request code

2. התכנית תקלוט מהמשתמש תו בין a ל i (המשתמש יכניס תו ואחריו ילחץ enter), כאשר התו מייצג פעולה אשר השרת צריך לבצע עבור הלקוח (ניתן להניח קלט תקין. יש לתמוך באותיות גדולות וקטנות).
3. אם המשתמש הכניס את התו i, יש לסיים את צד **הלקוח** בצורה נקיה (פינוי משאבים אשר אין בהם צורך, לא כולל הזיכרון המשותף).
   אחרת, התכנית שלכם צריכה לכתוב את התו **לתחילת הזיכרון המשותף**. שימו לב כי הכתיבה לזיכרון המשותף צריכה להיעשות רק כאשר השרת כבר קרא את המידע שיש שם. אם יש בזיכרון המשותף תו אשר השרת לא קרא **יש להמתין עם הכתיבה (לא לבצע busy waiting)**.
4. חזור שוב לשלב מספר 1.

שימו לב: מספר לקוחות רוצים לכתוב תו לזיכרון המשותף במקביל, אתם צריכים לגרום להם להמתין עד שהשרת יקרא את המידע שיש שם ורק לאחר מכן לקוח אחד יכתוב את המידע שלו לזיכרון המשותף.

<u>הנחיות עבור ex42 – צד שרת</u>

- שם התרגיל: ex4
- שם קבצי מקור (source file) שיש לשלוח: ex42.c

על השרת לייצר קובץ בעל השם <*yourID*>.txt למטרת כתיבה וקריאה.
על השרת לייצר זיכרון משותף בעזרת מפתח ייחודי (שם הקובץ למפתח הוא <*yourID*>.txt, התו שתשתמשו בו נתון להחלטתכם).
לאחר יצירת הזיכרון המשותף על השרת לייצר מערך של threads בגודל 5, ותור דינאמי (Job queue) מסוג char אשר יכיל תווים המסמלים פעולות אותן ה threads צריכים לבצע.
בנוסף עליכם לייצר משתנה גלובלי שלם אשר שמו יהיה internal_count ולאתחלו לאפס.

כאשר מידע חדש נכתב לזיכרון המשותף על ה thread הראשי לבצע את הפעולות הבאות:

1. לקרוא את המידע מהזיכרון המשותף
2. להוסיף ל job queue את התו החדש אשר הוא קרא (רק אם התו שנקרא בין a ל f)
3. להמתין למידע חדש שייכתב לזיכרון המשותף.

להלן הפעולה אותה צריך לבצע thread מה thread pool:
עבור התו a : הגרלת מספר X בין 10 ל 100, על ה thread להיכנס למצב שינה של X ננו שניות בעזרת קריאת המערכת **nanosleep**, לאחר השינה עליו להוסיף 1 ל internal_count
עבור התו b : הגרלת מספר X בין 10 ל 100, על ה thread להיכנס למצב שינה של X ננו שניות בעזרת קריאת המערכת **nanosleep**, לאחר השינה עליו להוסיף 2 ל internal_count
עבור התו c : הגרלת מספר X בין 10 ל 100, על ה thread להיכנס למצב שינה של X ננו שניות בעזרת קריאת המערכת **nanosleep**, לאחר השינה עליו להוסיף 3 ל internal_count
עבור התו d : הגרלת מספר X בין 10 ל 100, על ה thread להיכנס למצב שינה של X ננו שניות בעזרת קריאת המערכת **nanosleep**, לאחר השינה עליו להוסיף 4 ל internal_count
עבור התו e : הגרלת מספר X בין 10 ל 100, על ה thread להיכנס למצב שינה של X ננו שניות בעזרת קריאת המערכת **nanosleep**, לאחר השינה עליו להוסיף 5 ל internal_count
עבור התו f: על ה thread להוסיף לקובץ <*yourID*>.txt את המזהה הייחודי של ה thread ואת הערך שיש בתוך internal_count ולרדת שורה. לדוגמא:
thread identifier is 3882 and internal_count is 5

כאשר ה thread הראשי קורא מהזיכרון המשותף את התו g עליו לסיים <u>באותו רגע</u> את פעולתם של שאר ה threads ולאחר שסיימו את פעולתם עליו להוסיף לקובץ <*yourID*>.txt, את המזהה הייחודי שלו ואת הערך שיש בתוך internal_count ולרדת שורה בפורמט שניתן מקודם.
לאחר הכתיבה לקובץ עליו לנקות את משאבי המערכת אותם הקצה (לא כולל הזיכרון המשותף), כמובן שהוא לא צריך למחוק את הקובץ <*yourID*>.txt.

כאשר ה thread הראשי קורא מהזיכרון המשותף את התו h עליו להמתין שכל העבודות אשר נמצאות באותו רגע ב job_queue יבוצעו (כלומר כל העבודות שמגיעות לאחר קבלת התו h ייכנסו לתור אך לא ייספרו כעבודות שה thread צריך להמתין להם). לאחר ביצוע כל העבודות עליו לגרום לכל חמשת ה threads לסיים את פעולתם כאשר לפני מותו של כל thread עליו להוסיף לקובץ <*yourID*>.txt את המזהה הייחודי שלו ואת הערך שיש בתוך internal_count ולרדת שורה בפורמט שניתן מקודם.
לאחר כתיבת המידע של כל ה threads ולאחר סיום/מוות של כל ה thread , על ה thread הראשי להוסיף לקובץ <*yourID*>.txt את המזהה הייחודי שלו ואת הערך שיש בתוך internal_count ולרדת שורה בפורמט שניתן קודם. לאחר הכתיבה לקובץ עליו לנקות את כל משאבי המערכת אותם הקצה (זיכרון משותף/סמפורים), כמובן שהוא לא צריך למחוק את הקובץ <*yourID*>.txt.

<u>הערות:</u>

1. **אסור לתהליך השרת לכתוב מידע לזיכרון המשותף, הוא צריך רק לקרוא משם מידע.**
2. **שימו לב: לשרת אסור להשתמש במנגנוני IPC על מנת לעדכן את תהליכי הלקוח שהוא קרא את התו שיש בזיכרון המשותף (רמז: שימוש בסמפורים).**
3. **שימו לב כי מספר threads יכולים לכתוב במקביל לקובץ הפלט ולכן עליכם לסנכרן פעולה זו.**
4. **אתם צריכים להחליט לבד כמה סמפורים וכמה מיוטקסים אתם צריכים ובאילו תווים תשתמשו בשביל לייצר את המפתח לזיכרון המשותף ולסמפורים שלכם.**
5. **שימו לב שבמידה ואחד הלקוחות הכניס את התו g או h, על הלקוח לנקות את משאבי המערכת הרלוונטיים אליו, אך אין לכך שום השפעה על פעולתו של השרת.**
6. **שימו לב שעל מנת שתוכלו להריץ את התוכנית שלכם מספר פעמים על ה U2, עליכם להקפיד לנקות (לשחרר את הסמפורים/מיוטקסים שהקצתם) – עבודה לא נכונה תגרום למצב שבו לא תוכלו להריץ את התרגיל על ה U2.**

בהצלחה!

☺☺☺☺☺☺☺☺☺☺☺☺☺☺