# מכללה אקדמית הדסה
## החוג למדעי המחשב

## תרגיל #4 : תכנות מערכת ומבוא לתכנות מקבילי-- צינור משוים ותור הודעות

בחלק מתרגיל זה נחזור למשימה מתרגיל 3b, כדי להתרשם מההבדלים בין pipe, named pipe, message queue. נקווה שתוכלו להתרשם עד כמה תור הודעות מקל על ביצוע המשימה, בעוד עת התהליכים אינם הורה וילדיו, צינור משוים נעשה מסורבל.

**תכנית a – חזרה על תכנית b מתרגיל #3– עם named pipe**

כתבו את התכנית הבאה:

תהליך א' (שהיה בתרגיל 3# האב), וייקרא עתה ממלא המערך, מגדיר מערך של מספרים שלמים בן ARR_SIZE (קבעוהו להיות 100) תאים. מטרתו של תהליך א' היא למלא את המערך במספרים ראשוניים הולכים וגדלים.

בתכנית זאת תהליך א' (הממלא) לא מוליד ילדים, אלא אתם מרצים בעצמכם NUM_OF_GEN (קבעוהו להיות 3) תהליכים של תכנית שניה שתכתבו, ותקרא כאן: יצרן המספרים. יצרן המספרים בלולאה אינסופית מגריל מספרים בתחום $2..10^6$, ועת מוגרל על-ידו מספר ראשוני שולחו לממלא המערך יחד עם המזהה שלו (דרך named pipe). ממלא המערך צובר את המספרים הדרושים לו במערך (אלה שגדולים או שווים מקודמיהם). הממלא, בכל סיבוב בלולאה הראשית שלו, קורא מספר מהתור המשוים שלו. הוא בודק האם המספר יכול להתווסף למערך, ושולח ליצרן ששלח לו את המספר, פידבק 0 או 1.

עת הממלא גומר למלא את המערך (או נכשל מאה פעמים בהוספת ערך חדש מערך) הוא לא הורג את היצרנים, כי הם לא ילדיו, אלא הוא שולח להם הודעה שמורה להם לסיים, ומציג אותו פלט כמו בתרגיל הקודם: מספר התאים בהם יש ערכים במערך, הערך המזערי, הערך המירבי. עת יצרן מספרים מתבקש ע"י הממלא לסיים הוא מציג כמה ערכים שהוא הגריל נוספו למערך.

שימו לב שייתכן מצב בו יצרן שלח מספר לממלא, הממלא כבר לא זקוק למספר זה (כי המערך שלו מלא), ולכן הוא יישלח ליצרן משוב, למשל, 1-, שיגרום ליצרן לסיים.

כדי למנוע מצב בו היצרן הראשון בלבד 'זוכה' לשלוח מספרים לממלא המערך (טרם ששני היצרנים האחרים בכלל התחילו לרוץ) נקבע כי: כל יצרן בתחילת ריצתו שולח לממלא המערך את מספרו (1, 2, או 3). הממלא אחרי שקרא שלושה מספרים משלושה יצרנים מסיק שכל היצרנים רצים. הוא יישלח להם חזרה מספר שמורה להם: 'עתה אתם רשאים להתחיל להעביר לי מספרים'. כל יצרן, כאמור, שולח לממלא את מספרו, קורא מהממלא מספר, ורק אז פונה ללולאה המרכזית שלו, בה הוא מגריל מספרים, שולחם, ומקבל פידבק.

הערות

א. כל יצרן מספרים יקבל דרך וקטור הארגומנטים את שם התור של בעל המערך, ואת המספר שלו (1, 2, 3).
ב. התור של כל יצרן ייקרא fifoX אם X הוא המספר שלו.
ג. כל יצרן יוכל לשלוח לממלא המערך לצד המספר שהוא הגריל גם את X, והממלא יחזיר לו תשובה דרך התור המתאים.
ד. X גם ישמש כ: seed של כל יצרן.
ה. שמות הקבצים: ex4a1.c, ex4a2.c

נגדיר בצורה ברורה את הארגומנטים.

תכנית א' תורץ:

./ex4a1 fifo0 fifo1 fifo2 fifo3

כאשר תכנית א' קוראת מ: fifo0

תכנית ב' תורץ:

./ex4a2 fifo0 1

(ובהתאמה 2 או 3 לעותק השני או השלישי)

**<u>תכנית b – חזרה על תכנית b מתרגיל #3 – עם message queue</u>**

כתבו אותה תכנית כמו בסעיף a מעל, אולם עם תור הודעות אותו יקצה ממלא המערך.

<u>הערות</u>

א. לצורך ייצור המספרים האקראיים השתמשו ב: seed : 1, 2, 3 (לפי מספר היצרן). ערך זה יועבר ליצרן דרך וקטור הארגומנטים
ב. זמנו: ; ('4', ".")ftok
ג. שמות הקבצים: ex4b1.c, ex4b2.c

**<u>תכנית c</u>**

כתבו את שלוש התכניות הבאות:

<u>שרת ראשוניות</u>

עם תחילת ריצתו פותח תור הודעות, באמצעותו יפנה אליו לקוח המעוניין בשרותיו.

השרת רץ בלולאה אינסופית. בכל סיבוב בלולאה הוא קורא הודעה הכוללת מחרוזת המורכבת מסדרה של מספרים טבעיים (<= 2) מופרדים זה מזה ברווח אחד בדיוק (מחרוזת כגון: "2 3897 100 17"). הוא ממיר כל מספר המופיע במחרוזת לערך מספרי, ובודק את ראשוניותו. הוא מחזיר ללקוח שפנה אליו מחרוזת הכוללת את המספרים הראשוניים שהופיע בהודעה שהוא קיבל.

<u>שרת פלינדרומיות.</u>

מתנהל באופן דומה. עם תחילת ריצתו פותח תור הודעות, באמצעותו יפנה אליו לקוח המעוניין בשרותיו.

השרת רץ בלולאה אינסופית. בכל סיבוב בלולאה הוא קורא הודעה הכוללת מחרוזת (מחרוזת כגון: "axyxa"). בודק האם המחרוזת היא פלינדרום ומחזיר תשובה כן\לא.

.

<u>תהליך front end</u>

בלולאה אינסופית:

(א) קורא מהמשתמש את הערך p ואחריו סדרת מספרים שיש לבדוק את ראשוניותם עד קריאת הערך אפס שמסמן את סוף הסדרה, או את הערך q מחרוזת שאת הפלינדרומיות שלה יש לבדוק.

(ב) שולח לשרת המתאים את ההודעה המתאימה, מחכה ממנו לתשובה ומדפיס, במקרה של בדיקת ראשוניות את המספרים שחזרו מהשרת, ובמקרה של בידקת פלינדרומיות את הפלט + או -.

כל תכנית תסתיים ע"י קבלת הסיגנל SIGINT. ב: signal handler ישוחררו תורי ההודעות שהוקצו.

נקבע של: ftok נעביר את ".", וכן את התווים: p, q.

<u>שמות הקבצים לפי הקונבנציות שקבענו בתחילת הסמ':</u>
ex4c1.c = שרת ראשוניות.
ex4c2.c = שרת פלינדרומיות
ex4c3.c = front end

<u>שאלה</u>: האם השרתים הללו יוכלו לקבל פניה מכמה תהליכי front end שירוצו במקביל?