

## Document Design-exe4

- קישור ל-GitHub:

<https://github.com/orelklodintwito/Systems-Programming-Assignment-4.git>

- מטרת התוכנית:

מטרת התוכנית היא לממש מערכת Echo מבוססת TCP בשפת C, הכוללת שרת רבת-הליכוני ותוכנית לקוח המדמה ריבוי חיבורים במקביל. השרת מאזין לחיבורים נכנסים, יוצר תהליכון ייעודי עבור כל לקוח, מקבל נתונים, ממיר אותיות קטנות לגדולות ושולח את התוצאה בחזרה ללקוח.

בנוסף, השרת מנהל משתנה גלובלי הסופר את מספר הלקוחות המחוברים בו-זמנית, כאשר הגישה אליו מתבצעת בתוך קטע קריטי המוגן באמצעות mutex לצורך מניעת התנגשות בין תהליכונים. תוכנית הלקוח יוצרת מספר תהליכונים הפועלים במקביל, כאשר כל אחד מהם מתחבר לשרת, שולח הודעה, מקבל תשובה ומסיים את פעולתו.

המימוש מדגים עבודה עם System Calls בסביבת Unix, שימוש בסוקטים לתקשורת TCP, ניהול תהליכונים וטיפול ב-partial send/receive בהתאם לדרישות היעילות והסנכרון של המטלה.

- כיצד עומדת בדרישות:

התוכנית עומדת בדרישות המטלה באמצעות מימוש ישיר של ארכיטקטורת שרת-לקוח מבוססת TCP בשילוב ריבוי תהליכונים וסנכרון משאבים. השרת מאזין לחיבורים נכנסים, ובכל חיבור חדש נוצר תהליכון ייעודי המטפל בלקוח באופן עצמאי, בהתאם לדרישת שרת רבת-הליכוני. כל תהליכון מקבל נתונים מהלקוח, ממיר אותיות קטנות לגדולות ושולח את התוצאה חזרה, ובכך מממש את מנגנון ה-Echo הנדרש.

השרת מנהל משתנה גלובלי הסופר את מספר הלקוחות המחוברים בו-זמנית. הגישה למשתנה זה מתבצעת בתוך קטע קריטי המוגן באמצעות mutex, ובכך נמנעים מצבי race condition בין תהליכונים, כנדרש בדרישות הסנכרון.

התוכנית עושה שימוש ב-buffer-פנימי בגודל 4096 בתים לצורך קבלת ושליחת נתונים. פעולות השליחה והקבלה ממומשות באמצעות לולאות המבטיחות טיפול במצבי partial send/receive, שכל הנתונים מועברים במלואם גם כאשר קריאות המערכת מחזירות חלק מהמידע בלבד.

תוכנית הלקוח מדמה ריבוי משתמשים באמצעות יצירת מספר תהליכונים הפועלים במקביל. כל תהליכון יוצר socket מתחבר לשרת, שולח הודעה, מקבל תגובה ומסיים את פעולתו תוך סגירת משאבים תקינה.

בנוסף, כל קריאת מערכת נבדקת, ובמקרה של שגיאה מתבצע טיפול מתאים הכולל הדפסת הודעת שגיאה וניקוי משאבים. בכך נשמרת יציבות המערכת ועמידה בדרישות היעילות והאמינות של המטלה.

• ניתוח קריאות מערכת (System Calls):

קריאת מערכת	מתי נקראת בתוכנית	ערך חזרה	תפקידה בתוכנית
socket	בעת יצירת socket בשרת ובלקוח	מזהה socket או -1 בשגיאה	יצירת נקודת תקשורת TCP
bind	באתחול השרת	0 בהצלחה או -1 בשגיאה	שייך ה socket-לכתובת ופורט
listen	לאחר bind בשרת	0 בהצלחה או -1 בשגיאה	העברת השרת למצב האזנה לחיבורים
accept	בלולאת השרת	מזהה socket חדש או -1 בשגיאה	קבלת חיבור מלקוח
connect	בתוכנית הלקוח	0 בהצלחה או -1 בשגיאה	חיבור הלקוח לשרת
send	בעת שליחת נתונים	מספר בתים שנשלחו או -1 בשגיאה	העברת נתונים דרך socket
recv	בעת קבלת נתונים	מספר בתים שהתקבלו או -1 בשגיאה	קבלת נתונים מהצד השני
close	בסיום שימוש ב socket	0 בהצלחה או -1 בשגיאה	סגירת חיבור ושחרור משאבים

התוכנית עושה שימוש בקריאות מערכת מרכזיות של Unix לצורך תקשורת רשת מבוססת TCP וניהול חיבורים בין שרת ללקוחות. קריאת socket משמשת ליצירת נקודת תקשורת, ולאחר מכן השרת מבצע bind ו listen-כדי לאפשר קבלת חיבורים נכנסים. כל חיבור חדש מתקבל באמצעות accept ונמסר לטיפול תהליכון ייעודי.

בתוכנית הלקוח נעשה שימוש ב connect-לצורך יצירת חיבור לשרת. העברת הנתונים מתבצעת באמצעות send ו recv-כאשר לולאות משלימות מבטיחות טיפול במצבי partial send/receive.

בסיום העבודה, קריאת close משמשת לסגירת ה socket-ושחרור משאבים. כל קריאת מערכת נבדקת לפי ערך החזרה שלה, ובמקרה של שגיאה מתבצע טיפול מתאים הכולל הדפסת הודעת שגיאה וניקוי משאבים, בהתאם לדרישות המטלה.

• **לוגיקת זרימה (Flow):**

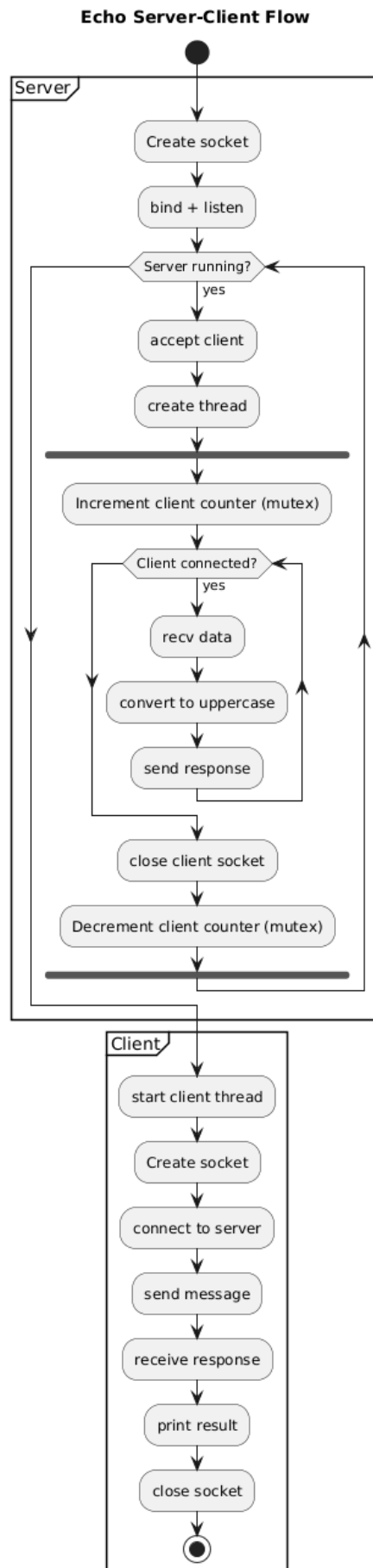
התוכנית מממשת מודל שרת-לקוח מבוסס TCP, כאשר השרת פועל באופן רציף לקבלת חיבורים חדשים, והלקוח מדמה מספר חיבורים במקביל. זרימת הפעולה מתחלקת לשני חלקים: פעולת השרת ופעולת הלקוח.

**זרימת השרת**

1. אתחול השרת והכנת socket:  
השרת יוצר socket TCP באמצעות socket, מגדיר אפשרויות תקשורת, משייך אותו לכתובת ולפורט באמצעות bind ומעביר אותו למצב האזנה באמצעות listen.
2. המתנה לחיבורים נכנסים:  
השרת נכנס ללולאה אינסופית וממתין לחיבורי לקוחות באמצעות accept. כל חיבור חדש מתקבל ומזוהה באמצעות socket ייעודי.
3. יצירת תהליכון לטיפול בלקוח:  
עבור כל לקוח חדש נוצר תהליכון באמצעות pthread\_create. התהליכון מקבל את מזהה ה socket ומטפל בלקוח באופן עצמאי.
4. עדכון מונה לקוחות:  
עם תחילת הטיפול בלקוח, התהליכון מגדיל מונה גלובלי של לקוחות מחוברים בתוך קטע קריטי המוגן באמצעות mutex.
5. קבלת נתונים מהלקוח:  
התהליכון קורא נתונים מהלקוח באמצעות recv בלולאה, תוך טיפול במצבי partial receive.
6. עיבוד הנתונים:  
הנתונים שהתקבלו עוברים המרה מאותיות קטנות לגדולות באמצעות פונקציית עיבוד ייעודית.
7. שליחת התגובה ללקוח:  
הנתונים המעובדים נשלחים חזרה ללקוח באמצעות send בלולאה המבטיחה שליחה מלאה.
8. סיום חיבור הלקוח:  
כאשר הלקוח מתנתק או מתרחשת שגיאה, ה socket נסגר, מונה הלקוחות מתעדכן בתוך קטע קריטי, והתהליכון מסתיים.

**זרימת הלקוח**

1. יצירת תהליכונים:  
תוכנית הלקוח יוצרת מספר תהליכונים הפועלים במקביל, כאשר כל תהליכון מייצג לקוח עצמאי.
2. יצירת חיבור לשרת:  
כל תהליכון יוצר socket ומתחבר לשרת באמצעות connect.
3. שליחת הודעה:  
הלקוח שולח הודעת טקסט לשרת באמצעות send בלולאה המבטיחה שליחה מלאה.
4. קבלת תשובה:  
הלקוח מקבל את תגובת השרת באמצעות recv ומדפיס את הנתונים שהתקבלו.
5. סגירת חיבור:  
לאחר השלמת התקשורת, ה socket נסגר והתהליכון מסתיים.



**ניהול Buffer ועילות :**

התוכנית עושה שימוש ב־Buffer פנימי בגודל 4096 bytes הן בצד השרת והן בצד הלקוח, בהתאם לדרישות המטלה. בחירה זו מאפשרת טיפול יעיל בנתונים תוך שמירה על איזון בין שימוש בזיכרון לבין מספר קריאות מערכת.

גודל Buffer של 4096 תואם לגודל נפוץ של עמוד זיכרון (Page) במערכות הפעלה, ולכן מאפשר עבודה יעילה עם זיכרון והעתקות נתונים. בנוסף, גודל זה מאפשר לקבל ולשלוח כמות נתונים משמעותית בכל קריאה אחת של recv/send, ובכך להפחית את מספר ה־System Calls הנדרשות ביחס לבאפר קטן יותר.

מבחינת יעילות, התוכנית מצמצמת את מספר קריאות המערכת בכך שהיא :

- מבצעת recv לקבלת נתונים ב־chunks עד גודל ה־buffer במקום לקרוא תורו או במקטעים קטנים.
  - מבצעת send באמצעות פונקציה ייעודית (send\_all) המשלימה שליחה מלאה בלולאה רק במקרה שבו send מחזיר שליחה חלקית (partial send). כך נמנעת חזרה מיותרת על קריאות send כאשר אין בכך צורך.
  - בצד הלקוח מתקבלת התגובה בגודל הנדרש בלבד, בהתאם לאורך ההודעה שנשלחה, ובכך נמנעת קריאה עודפת.
- במקרה של partial operations (כאשר recv/send מחזירות פחות בתים מהמבוקש), נעשה שימוש בלולאות המבטיחות השלמת הפעולה במלואה. גישה זו מבטיחה תקינות תקשורת ומונעת אובדן מידע, תוך שמירה על מינימום קריאות מערכת: הלולאות רצות רק כאשר יש צורך בכך, כלומר רק כאשר הקריאה לא הושלמה בפעם אחת.

**• טיפול בשגיאות:**

התוכנית כוללת מנגנון שיטתי לזיהוי וטיפול בשגיאות בכל קריאות המערכת (System Calls), הן בצד השרת והן בצד הלקוח. כל קריאה קריטית נבדקת לפי ערך החזרה שלה, ובמקרה של כשל מתבצע טיפול מתאים שמטרתו לשמור על יציבות המערכת ולמנוע דליפות משאבים.

בעת יצירת socket, חיבור (connect), קבלת חיבורים (accept), שליחת נתונים (send) וקבלת נתונים (recv), נבדק האם ערך החזרה מצביע על שגיאה. במקרה כזה מודפסת הודעת שגיאה באמצעות perror, המאפשרת זיהוי ברור של מקור התקלה. לאחר מכן מתבצע ניקוי משאבים רלוונטיים, כגון סגירת socket או שחרור זיכרון, לפני סיום הפעולה או חזרה ללולאת ההמתנה.

בקריאות recv/send מתבצע טיפול גם במצב שבו הקריאה נקטעת על ידי אות מערכת (EINTR). במקרה זה הקריאה מתבצעת מחדש במקום להיחשב כשגיאה, וכך נשמרת רציפות הפעולה. בנוסף, זיהוי ערך חזרה של 0 בקריאת recv מפורש כסיום חיבור מצד הלקוח, והחיבור נסגר בצורה תקינה.

בשרת, במקרה של כשל ביצירת תהליכון (pthread\_create) או הקצאת זיכרון, החיבור ללקוח נסגר כדי למנוע שימוש במשאבים לא תקינים. מונה הלקוחות הגלובלי מעודכן תמיד בתוך קטע קריטי, גם במצבי סיום חריגים, כדי לשמור על עקביות הנתונים.

גישה זו מבטיחה שהתוכנית מתמודדת עם כשלים בצורה מבוקרת, שומרת על משאבי המערכת וממשיכה לפעול ככל האפשר, בהתאם לדרישות האמינות והיציבות של המטלה.