

## Document Design-exe4

- **קישור ל-GitHub :**

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

- **מטרת התוכנית:**

מטרת התוכנית היא למשם מערכת Echo מבוססת TCP בשפת C, הכוללת שרת ורשת תקשורת ותוכנית ללקוח המדמה ריבוי חיבורים במקביל. השרת מאזין לחיבורים נכנים, יוצר תהליכי ייוזדי עבור כל לקוח, מקבל נתונים, ממיר אותן קטנות לגודלות ושולח את התוצאה בחזרה ללקוח.

בנוסף, השרת מנהל משטנה גלובלי הסופר את מספר הלקוחות המוחברים בויזמנית, כאשר הגישה אליו מתבצעת בטור קטע קרייטי המונע באמצעות mutex לצורך מניעת התנגשות בין תהליכיים. תוכנית הלקוח יוצרת מספר תהליכי פעולה בסוקטים במקביל, כאשר כל אחד מהם מתחבר לשרת, שולח הודעה, מקבל תשובה ומסיים את פעולתו. השימוש מדגים עבודה עם System Calls בסביבת Unix שימוש בסוקטים לתקשורת partial send/receive ניהול תהליכיים וטיפול ב-

המטלה.

- **כיצד עומדת בדרישות:**

התוכנית עומדת בדרישות המטליה באמצעות מימוש ישיר של ארכיטקטורת שרת-לקוח מבוססת TCP בשילוב ריבוי תהליכיים וסנכרון משאים. השרת מאזין לחיבורים נכנים, ובכל חיבור חדש נוצר תהליך ייוזדי המטפל בלקוח באופן עצמאי, בהתאם לדרישת שרת ורשת תקשורת. כל תהליך מקבל נתונים מהלקוח, ממיר אותן קטנות לגודלות ושולח את התוצאה חזרה, ובכך מימוש את מגנון ה Echo-הנדרש.

השרת מנהל משטנה גלובלי הסופר את מספר הלקוחות המוחברים בויזמנית. הגישה למשטנה זה מתבצעת בטור קטע קרייטי המונע באמצעות mutex ובכך מנעים מצב race condition בין תהליכיים, כנדרש בדרישות הסנכרון.

התוכנית עשויה שימוש ב-buffer פנימי בגודל 4096 בתים לצורך קבלת ושליחת נתונים. פעולות השליחה והקבלה ממושכות באמצעות לולאות המבטיחות טיפול במצב partial receive/send, כך שכל הנתונים מועברים במלואם גם כאשר קריאות המערכת מחזירות חלק מה מידע בלבד.

התוכנית הלקוח מדמה ריבוי משתמשים באמצעות יצירת מספר תהליכי פעולה במקביל. כל תהליך יוצר socket, מתחבר לשרת, שולח הודעה, מקבל תגובה ומסיים את פעולתו תוך סגירת משאים תקינה.

בנוסף, כל קריאת מערכת נבדקת, ובמקרה של שגיאה מתבצע טיפול מותאים הכולל הדפסת הודעה שגיאה וניקוי משאים. בכך נשמרת יציבות המערכת ועמידה בדרישות הייעילות והאמינות של המטליה.

• **ניתוח קריואות מערכת (System Calls):**

תקמידה בתוכנית	ערך חזרה	מתי נקראת בתוכנית	קריית מערכת
יצירת נקודת תקשורת TCP	מספר socket או -1 בשגיאה	בעת יצירת socket בשרות ובלוקה	socket
שיוך ה socket-לכתובת ופורט	-1 בבכלה או בשגיאה	באתחול השרת	bind
העברת השרת למצב האזנה לchipovers	-1 בבכלה או בשגיאה	לאחר bind בשרות	listen
קבלת חיבור מלוקה	מספר socket חדש או ובשגיאה	בלולאת השרת	accept
חיבור הלוקה לשרת	-1 בבכלה או בשגיאה	בתוכנית הלוקה	connect
העברת נתונים דרך socket	מספר בתים שנשלחו או -1 בשגיאה	בעת שליחת נתונים	send
קבלת נתונים מהצד השני	מספר בתים שהתקבלו או -1 בשגיאה	בעת קבלת נתונים	recv
סגירת חיבור ושחרור משאים	0 בהצלחה או -1 בשגיאה	בסיום שימוש ב-socket	close

התוכנית עשויה שימוש בקריאות מערכת מרכזיות של `socket` תקשורת רשת מבוססת TCP וניהולchipovers בין שרת לлокחות. קריית socket משמשת ליצור נקודת תקשורת, ולאחר מכן השרת מבצע `bind` ו-`listen` לאפשר קבלתchipovers נכנים. כל חיבור חדש מתקבל באמצעות `accept` וונומסר לטיפול תחילכו ייעודי.

בתוכנית הלוקה 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, קריית `close` משמשת לסגירת ה socket-ושחרור משאים. כל קריית מערכת נבדקת לפי ערך החזרה שלה, ובמקרה של שגיאה ממבצע טיפול מותאים כולל הדפסת הודעה שגיאה וניקוי משאים, בהתאם לדרישות המטלה.

• **לוגיקת זרימה (Flow):**

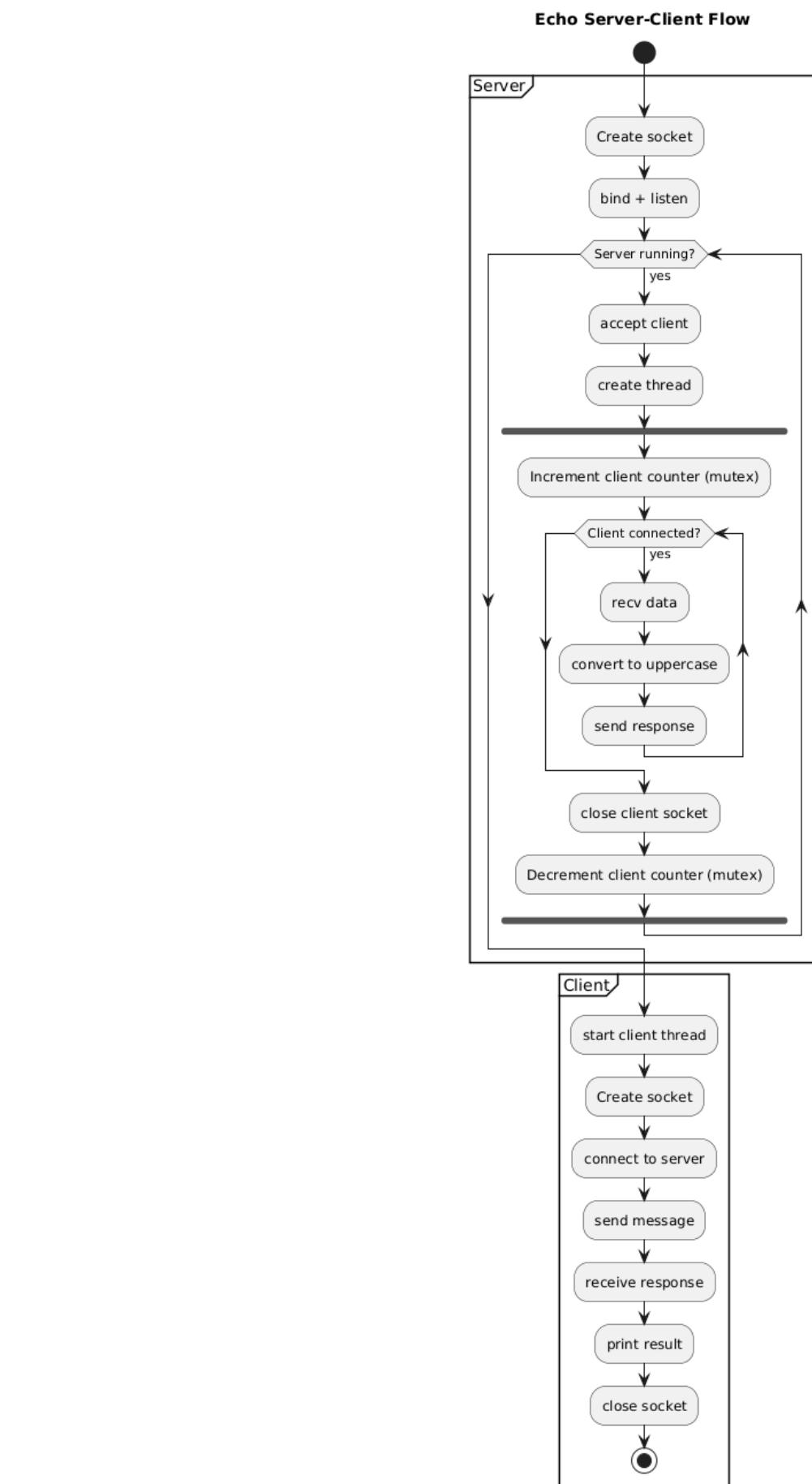
התוכנית מימושת מודל שרת-לקוח מבוסס TCP כאשר השירות פועל באופן רציף לקבלת חיבורים חדשים, והלקוח מדמה מספר חיבורים במקביל. זרימת הפעולה מתחלקת לשני חלקים: פועלות השירות ופעולות הלוקה.

**זרימת השירות**

1. **אתחול השירות והכנת socket:** השירות יוצר socket TCP באמצעות מוגדר אפשרויות תקשורת, משייך אותו לכתובת ול포רט באמצעות bind ומעברו למצוות listen.
2. **המתנה לחיבורים נכנים:** השירות נכנס ללולאה אינטנסיבית וממתין לחיבור ללקוח באמצעות accept. כל חיבור חדש מתקבל ומוזהה באמצעות socket ייעודי.
3. **יצירת תהליכי לטיפול בלוקות:** עבור כל לקוח חדש נוצר תהליכי בטיפול באמצעות pthread\_create. התהליכי מתקבלים מזיהה ה socket-ומטפל בלוקוח באופן עצמאי.
4. **עדכו מונה לקוחות:** עם תחילת הטיפול בלוקות, ה手続き מגדיל מונה גלובלי של לקוחות מחוברים בתוך קטע קרייטי המוגן באמצעות mutex.
5. **קבלת נתונים מהлокות:** ה手続き קורא נתונים מהлокוח באמצעות recv בlolalla, תוך טיפול במצב receive.
6. **יעוב הנתונים:** הנתונים שהתקבלו עוברים המרה מאותיות קטנות לגודלות באמצעות פונקציית יעובוד ייעודית.
7. **שליחת התגובה ללוקות:** הנתונים המעובדים נשלחים חוזה ללקוח באמצעות send בlolalla המבטיחה שליחה מלאה.
8. **סיום חיבור הלוקות:** כאשר הלוקה מתנתק או מתרחשת שגיאה, ה socket-נסגר, מונה הלוקות מתעדכן בהתאם קטע קרייטי, וה手続き מסתיים.

**זרימת הלוקות**

1. **יצירת תהליכי:** תוכנית הלוקה יוצרת מספר תהליכי הפעולים במקביל, כאשר כל תהליכי מייצג לקוח עצמאי.
2. **יצירת חיבור לשרת:** כל תהליכי יוצר socket ומחבר לשרת באמצעות connect.
3. **שליחת הודעה:** הלוקה שולח הודעה טקסט לשרת באמצעות send בlolalla המבטיחה שליחה מלאה.
4. **קבלת תשובה:** הלוקה מקבל את תגובת השירות באמצעות recv ומדפיס את הנתונים שהתקבלו.
5. **סגירת חיבור:** לאחר השלמת התקשרות, ה socket-נסגר וה手続き מסתיים.



### **ניהול Buffer ויעילות:**

התוכנית עושה שימוש ב-`Buffer` פנימי בגודל 4096 bytes הן מצד השרת והן מצד הלקוח, בהתאם לדרישות המטלה. בחירה זו מאפשרת טיפול יעיל בתנאים 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 אפשר עבודה יעילה עם זיכרון והתקנות נתונים. בנוסף, גודל זה מאפשר לקבלת ולשלוח כמות נתונים משמעותית בכל קריאה אחת של `recv/send`, ובכך להפחית את מספר ה- `System Calls` הנדרשות ביחס למבצע קטן יותר.

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

- מבצעת `recv` לקבלת נתונים ב-`chunks` עד גודל `buffer` במקום לקרוא תויתו או במקטעים קטנים.
- מבצעת `send` באמצעות פונקציה ייודית (`send_all`) המשלימה שליחה מלאה בלבדה רק במקרה שבו `send` מוחזר שליחת חלקית (partial send). כך נמנעת חזרה מיותרת על קריאות `send` כאשר אין בכך צורך.
- בצד הלקוח מתקיים התגובה בגודל הנדרש בלבד, בהתאם לאורך הודעה שנשלחה, ובכך נמנעת קריאה נוספת.

במקרה של `partial operations` (כאשר `recv/send` מוחזרות פחות בזמנים מהמשך), נעשה שימוש בלאות המבטיחות השלמת הפעולה במלואה. גישה זו מבטיחת תקינות תקשורת ומונעת אובדן מידע, תוך שמירה על מינימום קריאות מערכת: הלאות רצות רק כאשר יש צורך בכך, כלומר רק כאשר הקריאה לא הושלמה בפעם אחת.

### **טיפול בשגיאות:**

התוכנית כוללת מגנון שיטתי לזיהוי וטיפול בשגיאות בכל קריאות המערכת (System Calls), הן מצד השרת והן מצד הלקוח. כל קריאה קריטית נבדקת לפני ערך החזרה שלו, ובמקרה של כשל מבצע טיפול מתאים שטרתו לשמר על יציבות המערכת ולמנוע דיליפות 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 הודעה שגיאה באמצעות  `perror`, המאפשרת זיהוי ברור של מקור התקלה. לאחר מכן מבצע ניקוי משאים לרלוונטיים, כגון סגירות `socket` או שחרור זיכרון, לפני סיום הפעולה או חזרה לולאות ההמתנה.

בקריאות `recv/send` מבצע טיפול גם במצב שבו הקריאה נתקעת על ידי אותן מערכת (`EINTR`). במקרה זה הקריאה ממבצעת מחדש במקום להיחשב כשגיאה, וכך נשמרת רציפות הפעולה. בנוסף, זיהוי ערך חזרה של 0 בקריאה `recv` מפורש כסיום חיבור מצד הלקוח, והחיבור נסגר بصورة תקינה.

בשרות, במקרה של כשל ביצירת תהליך (`pthread_create`) או הקצת זיכרון, החיבור ללקוח נסגר כדי למנוע שימוש במשאבים לא תקינים. מונה הלקוחות הגלובלי מעודכן תמיד בתוך קטע קרייטי, גם במקרים סיום חריגים, כדי לשמור על עקבויות הנתונים.

גישה זו מבטיחת שהתוכנית מתמודדת עם שללים בצוותה מבודקה, שומרת על משאבי המערכת וממשיכה לפעול ככל האפשר, בהתאם לדרישות האמינות והיציבות של המטלה.