**מערכת הפעלה מטלה 3**

**שאלה 1 :**

|  |
| --- |
| התבקשנו לממש שרת שפותח תקשורת בTcp ומממש מחסן של אטומים מסוג carbon , oxygen ו hydrogen . השרת יקבל בקשות ע"י מחרוזת שמתקבלת מלקוח שמשמש כספק שמילא את מחסניו.   * השרת פתח socket להאזנה בפורט שהגיע לו כארגומנט וממתין בפונקציית select לפקודת connect מלקוח מסוים. * ברגע שנזהה התחברות נחלץ את המידע הרלוונטי ונוסיף למשתנים של השרת, לאחר מכן נמתין להתחברות נוספת וכן הלאה. * בצד הלקוח (supplier) נקבל את הפורט וhostname * נפתח socket ונשלח פקודת connect אל השרת. * נבקש מהמשתמש להכניס את האטומים שהוא מספק ונכניס אותם לרשימה, נשלח את המידע כמחרוזת. * נדפיס את כל המידע שנמצא במחסן לאחר קבלת האטומים החדשים. |
| הוראות הרצה:   1. להיכנס אל תיקיית tar1 ( לאחר בניית make רקורסיבי( 2. ./ atom\_warehouse <port\_tcp> <port\_udp> 3. ./atom\_supplier localhost <port\_tcp> |
| בדוגמה זו נשלח :  ADD CARBON 30 לא שלחתי hudrogen  ADD OXYGEN 10 |
|  |
| ניתן לראות שאחרי כל פקודה שהשרת מדבל הוא מדפיס סיכום מעודכן, לאחר מכן ממתין למידע נוסף. כאשר הלקוח שלח את כל המידע הרלוונטי כרגע התקשורת נסגרת. |
|  |

|  |
| --- |
| **שאלה 2 :**  הסבר הרצה:   1. להיכנס אל תיקיית tar2 ( לאחר בניית make רקורסיבי( 2. ./molecule\_supplier <port\_tcp> <port\_udp> 3. ./atom\_supplier localhost <port\_tcp> 4. ./molecule\_requester localhost <port\_udp> |
| * שדרגנו את השרת בשאלה 1 אל שרת חדש בשם molecule\_supplier כך שבנוסף לקבלת מלאי חדש על tcp יוכל לקבל בקשות upd ליצירת מולקולות. * השרת החדש מקבל 2 ארגומנטים (פורט tcp ואז פורט udp) |
| * יצרנו לקוח חדש שמטרתו לקבל בקשות מהטרמינל למולקולה מסוימת (חמצן, פחמן , גלוקוז , אלכוהול), הלקח יקבל כארגומנט את הפורט בו השרת מצפה לקבל חבילות udp. * פתחנו socket מסוג datagram על מנת להעביר חבילות על גבי udp ושלחנו את המולקולה הנדרשת ומספר המולקולות אל השרת. |
|  |
| * בחירת בקשה של יצירת מולקולת water |
|  |
| * בצד השרת קיבלנו את הבקשה, פיצלנו אותה לפי רווחים וזיהינו איזה בקשה למולקולה קיבלנו ומה הכמות המבוקשת. * נבחן את המלאי שיש בשרת ונראה האם ישנה כמות מספיקה של אטומים הנדרשים לבניית המולקולה. במידה וכן נחסיר את הכמות הנדרשת ונחזיר הודעת הצלחה, במידה ולא נחזיר הודעת שגיאה ונשאיר את המלאי כמו שהוא. |
|  |
| קבלת בקשה DELIVER WATER 1 בצד השרת |
|  |
| הדגמה של בקשה בה אין מספיק אטומים:  DELIVER GLUCOSE 1 |
|  |
| צד שרת |
|  |

**שאלה 3 :**

|  |
| --- |
| * הוראות הרצה: * להיכנס אל תיקיית tar3 ( לאחר בניית make רקורסיבי( * ./ drinks\_bar <port\_tcp> <port\_udp> * ./atom\_supplier localhost <port\_tcp> * ./molecule\_requester localhost <port\_udp> |
| * כעת נאפשר לשרת לקבל קלט מהמקלדת במקביל לקלט udp השולח בקשות ליצירת מולקולות ולקלט tcp השולח בקשות לאספקת אטומים למחסן. * הקלט מהמקלדת יהיה פקודה מדויקת של יצירת משקה הבנוי ממספר סוגי מולקולות בכמויות שונות. * את האפשרות לעבוד המקביל נעשה בעזרת הכנסת הfd של stdin אל הset שעליו השרת מסתכל כל הזמן ובוחן האם ישנו מידע חדש שהגיע. * נעזר במשתנה זמני שיקבל את מהספר האטומים הנוכחיים ונבחן לפי המתכון כמה משקאות מהסוג המבוקש ניתן להכין. |
|  |
| דוגמת הרצה בטרמינל: |
|  |

**שאלה 4** :

|  |
| --- |
| הוראות הרצה :   1. יש להיכנס אל תיקיית tar4 (לאחר בניית make רקורסיבי( 2. ./drinks\_bar -T <tcp\_port> -U <udp\_port> [-o <oxygen>] [-c <carbon>] [-h <hydrogen>] [-t <timeout>] 3. ./atom\_supplier -h <hostname> -p <port> 4. ./molecule\_requester -h <hostname> -p <port> |
| * התבקשנו לממש שעון שיסגור את השרת לאחר זמן מסוים בו אין אף פנייה אליו. * נממש זאת באמצעות alarm עם 20 שניות טיימר, כמו כן נבצע שינוי בסיגנל SIGALRM כך שבמקום לצאת מהתוכנית יקפוץ לפונקציה שסוגרת תחילה את 2 החיבורים ואז יוצאת מהתוכנית. * נעדכן את השעות ל 60 שניות נוספות בכל קבלת בקשה חדשה. |
| * כאשר הבקשה מהטרמינל של השרת: |
|  |
| כאשר הבקשה מהספק: |
|  |
| כאשר הבקשה מהלקוח המבקש מולקולות: |
|  |
| קטעי הקוד הרלוונטיים למימוש השעון: |
|  |
|  |
| * כמו כן נתאר את תהליך הוספת האופציות (דגלים), נאלץ את הקלט להגיע עם 2 ארגומנטים לפורט tcp ו udp. * על מנת לפרסר בצורה תקינה (תווים ומילים) נעזר בפונקציה getoptlong , נשלח לה מספר ארגומנטים: * Argc , argv , את רשימת הדגלים , struct שמכיל את הכינוי הנוסף כמילה ארוכה בה אפשר לכנות כל דגל . * נבחן איזה דגל הגיע ונעדכן בעזרת switch case את הארגומנט המתאים. * בעזרת משתנים בוליאנים נאלץ לקבל את הקלט של Udp ו tcp . |
|  |
|  |
| נתאר מספר דוגמאות הרצה:   1. ללא דגלים אופציונאלית 2. עם חלק מהדגלים האפציונאליים 3. עם כל הדגלים האופציונאליים. 4. כל הדגלים עם מילים ארוכות. 5. ללא דגל חובה. 6. לא לפי הסדר. |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| התאמת הלקוח והספק לעבוד עם אופציות: |
|  |
|  |

**שאלה 5** :

|  |
| --- |
| הוראות הרצה :   1. יש להיכנס אל תיקיית tar5 (לאחר בניית make רקורסיבי( 2. ./drinks\_bar [-T <tcp\_port> -U <udp\_port> | -s <UDS stream file path> -d <UDS datagram file path>] [-o <oxygen>] [-c <carbon>] [-h <hydrogen>] [-t <timeout>] 3. ./atom\_supplier [-h <hostname> -p <port> | -f <file path>] 4. ./molecule\_requester [-h <hostname> -p <port> | -f <file path>] |
| * בשאלה זו התבקשנו לאפשר תקשורת באמצעות UDS ביצענו זאת באופן הבא: * תחילה ביצענו אבחנה בין קבלת נתיב לקובץ כארגומנט או פורט וhostname תוך אילוץ להופעה של אחד משניהם וזריקת שגיאה כאשר ישנה הופעה של שניהם. * בעת קבלת דגלים -s ו-d חילצנו את כתובת הקובץ אותם אנחנו ממפים לתקשורת ונאתחל משתנים גלובליים לטובת בדיקה וניתוב התקשורת לפונקציות שונות כמו כן נבחן סתירה בארגומנטים כמתבקש. |
|  |
| * ניתוב התקשורת לפונקציות שונות כשמדובר בuds |
|  |
| * בפונקציה run\_server\_uds ניצור 2 sockets מסוג AF\_UNIX כאשר אחד ישמש עבוד datagram והשני עבור stream. * בשניהם נאתחל את המקום בזיכרון, נגדיר struct מסוג socketaddr\_un ונאתחל את השדות שלו בהתאמה לקלט שקיבלנו. לאחר מכן נבצע bind בין הfd של הקובץ שהתקבל בארגומנט לבין ה socket שנוצר על מנת שיהיה ניתן לכתוב אליו. * בstream נבצע גם listen על מנת שנוכל לקבוע תור בדומה לtcp ובdgram לא נעשה זאת. * החל מכאן העבודה היא זהה לקודם למעט השינוי ב fd ובstruct השונים שהותאמו לuds כבר בשלב בניית ה socket . |
|  |
|  |
| * בלקוחות ביצענו התאמה דומה בפיצול הארגומנטים ויצירת הsockets , יש לשים לב שלקוח אחד עובד רק מעל udp \ dgram ולקוח אחד מעל tcp\stream וזה יבחר על פי הארגומנטים שנבחרו. |
| כעת נציג מספר דוגמאות הרצה:   1. קלט חסר 2. קלט סותר 3. קלט של hostname + port 4. קלט של file כולל עבודה במקביל עם הלקוח והספק וכולל עצירת שעון. |
| 1. חסר -d |
| 1. יש גם tcp + udp וגם -s + -d |
| 1. בדיקה כי הדרישות הקודמות עדיין עובדות: |
|  |
|  |
| 1. כעת נציג קבלת תוספת מהספק , בקשה של מולקולה מהלקוח ובקשה של משקה מהטרמינל. |
|  |
|  |
|  |

**שאלה 6**:

|  |
| --- |
| * התבקשנו לנהל שמירה של המלאי בקובץ שיסונכרן בין כלל התהליכים של הברמנים. * במידה ולא הגיע הדגל -f נעבוד על פי הערכים המתקבלים באופציות כמו בסעיפים הקודמים (בעבודה על גבי פורטים וגם uds ) * במידה והתקבל הדגל -f והקובץ לא קיים ניצור אותו ונתחיל לסנכרן אותו בכל שינוי שיתבצע כך שכל ברמן אחר יראה ב"זמן אמת" את המידע העדכני על המלאי. * את הכתיבה לקובץ וקריאה ממנו נבצע בעזרת הפונקציה mmap באופן זה נוכל להקצות חלק מקובץ (או קובץ שלם) בינארי לטובת זיכרון. באופן זה נאפשר לכלל התהליכים לגשת לאותו קובץ ולהתעדכן במידע (ולעדכן בעצמם) * כמו כן עלינו לוודא שאין גישה כפולה לקובץ, כלומר שאין 2 ברמנים שונים שמוסיפים אטומים אך לא מעודכנים שהברמן האחר גם מוסיף ובעצם משאירים ערך שגוי בקובץ. את הסנכרון הזה נבצע באמצעות fcntrl בכך שבכל כתיבה וקריאה תחילה ננעל את הקובץ ואת נבצע את הפעולה. |
| * נציג כאן את הדרך לזיהוי האם הקובץ קיים כדי לדעת האם לסנכרן אליו או ממנו. |
|  |
| * נציג את חלק הקוד שמבצע נעילה לקובץ ומיפוי למידע כאשר קוראים מהקובץ אל הstruct של האטומים: |
|  |
|  |
| כעת נציג מספר מקרים לטובת בדיקה שהתרגילים הקודמים עדיין עובדים וגם כי האפשרויות הנוספות ממומשות כראוי:   1. הרצה עם uds כולל אופציות וללא file 2. הרצה עם uds כולל אופציות עם file חדש 3. הרצה עם uds כולל אופציות ו file קיים, הוספת של אטומים, בקשה של מולקולות ובקשה של משקאות. 4. בדיקה שאחרי סעיף 3 המלאי מתעדכן ע"פ הקובץ. 5. הרצה עם udp + tcp על מנת לבדוק שהאפשרויות הקודמות עדיין עובדות. 6. הרצת מספר ברמנים במקביל וביצוע פעולת תוספת אטומים באחד והצגה כי הקובץ מסונכרן ביניהם בזמן אמת. |
| 1. |
| 2. |
| 3. |
|  |
|  |
| 4.אחרי טעינה מחדש: |
| 5. |
|  |
|  |