תרגילים 4 – reactor pattern

תאריך הגשה: התרגיל יוגש עד ה 24/5/23 8:00 AM

הנחיות כלליות:

- התוכנית תכתב בשפת C או C++
- עליכם להגיש את כל קבצי המקור, ובנוסף לכך Makefile
- Makefile לא תקין, מכל סיבה שהיא, עלול להוריד את הציון דרמטית, עד ל 0
- ה Makefile יכיל לפחות את פקודות all שמקמפלת את התוצרים, ואת פקודת clean שמוחקת את תוצאות הקימפול של פקודת ה all וכן את כל ה kraft
- עליכם להגיש קובץ הסבר להפעלת המטלה, בטקסט חופשי ודוגמאות הרצה. שם הקובץ יהיה readme.txt - מוריד 10 נקודות במידה ולא
- המטלה עלולה להיבדק אוטומטית, ולכן אין להוסיף קבצים מיותרים, בפרט קבצים מקומפלים.
- את המטלה יש להגיש בקובץ **ZIP** (לא RAR/TAR/7ZIP) – מוריד 15 נקודות
- שם קובץ ההגשה (ה ZIP) יהיה כמספר ת.ז. של המגיש. במידה ומגישים בזוג, יש להגיש רק פעם אחת, ושם הקובץ יהיה ת.ז_ת.ז – מוריד 15 נקודות במידה ולא.
- לא ניתן לקבל הארכה. במקרה של סיבה מוצדקת (באישור המחלקה) ניתן להוריד את המטלה מחשבון כלל ההגשות של הסטודנט.
- יש להקפיד על שמות קבצי ההרצה, כפי שמופיע ב uasge עבור כל סעיף שבא נדרש קובץ הרצה.
- במידה ונדרשת פונציאונליות מעבר לדרישות בתרגיל (כגון ארוץ תקשורת נוסף, או API שלא צוין במפורש) , אתם רשאים לממש בכל דרך סבירה ויעילה, ולהסביר את הנעשה בקובץ ה readme.txt
- מומלץ להשתמש בקודים קודמים, ומקורות אחרים באינטרנט.
  במידה והשתמשתם באתר כלשהו – צרפו את הקישור לכך (בקובץ ההסבר)
  במידה והשתמשתם ב ChatGPT (כן, מותר !) , צרפו את מלל הבקשה המדויק שהעברתם לו
- ניתן ליצר קבצי עזר כרצונכם, אך בסופו של דבר, אני דורשים שיהיהו קבצי ההרצה ו/או הספריות כמו שהתבקשתם
- במידה וחסר לכם פרט במטלה, (כדוגמת מספר התהליכים ב threadpoll) עליכם לקבל החלטה לבד, ולנמק זאת בקובץ ה readme.txt. מומלץ להביא אסמכתא לנימוק, למשל אתר שממליץ על אסטרטגיה מסויימת

עליכם לממש צ'אט התומך במספר לקוחות לא מוגבל ,תוך שימוש במבנה ה reactor,
וכן בעזרת poll(2) או select(2) - לבחירתכם
את ההסבר ל API זה , וכן דוגמאות קוד ניתן למצו במדריך של ביג'.
https://beej.us/guide/bgnet/html/#slightly-advanced-techniques (פרק 7)

מומלץ בחום לקרא את כל המטלה בעיון ולבנות תוכנית פעולה, לפני שאתם ניגשים למימוש.

ספריית ה reactor.
עליכם לממש ליבה של תבנית עיצוב. reactor כלומר מנגנון אשר יקבל fileDescriptor של לקוח ופונקציה לביצוע כאשר ה fd "חם". ה reactor יתמוך בקבלת מספר גדול של fd וירוץ בטרד יחיד. במידה ואין קלט נמתין עד לקבלת ההודעה מכל אחד מה fd שממתינים עליהם, ויבצע את הפונקציה המתאימה ל fd. שם הספריה יהיה st_reactor.so

הריאקטור צריך לתמוך רק(!) בקריאה. (לא צריך לתמוך בכתיבה ו exception)

לפי ה design patter של ה reactor עליכם לבצע זאת כך:
א) אתם מיישמים מבנה נתונים אשר ימפה בין ה FD לבין הפונקציה
ב) אתם מיישמים את מנגנון ה selector, שהוא האחראי ע"י poll או select לבצעה האזנה על כל ה FD ולבצע את הפעולה שרושמה בטבלה של ה reactor
ג) מכיוון שה selector מקשיב ל FD מרובים, ויתכן ויותר מאחד יהיה "חם" בשלב מסויים, במקרה כזה סדר הטיפול לא משנה.

תוכלו להגדיר struct השומר את המבנים הפרטיים של הריאקטור באופן הרצוי לכם.

ה API של הספרייה יהי כך:
א.
Void* createReactor()
שיוצר ריאקטור.
מחזיר מצביע למבנה הריאקטור שיועבר לפונקציות הבאות. כאשר הריאקטור נוצר הוא לא עובד אבל כל מבני הנתונים יאולקצו.(init and alloc)
ב.
Void stopReactor(void * this)
עוצרת את הריאקטור אם הוא פעיל. אחרת לא עושה דבר.
ג.
Void startReactor(void * this)
מתחיל טרד של ריאקטור.
הטרד יחיה בביזי לופ ויקרא בפועל ל select או poll

ד. Void addFd (void * this,int fd, handler_t handler)
כאשר handler_t הוא פוינטר לפונקציה שתקרא כאשר ה fd חם.
תוכלו לעשות typedef ל handler_t איך שתרצו.)

ה.
Void WaitFor(void * this)
המתן על pthread_join(3) עד שהטרד של הריאקטור יסתיים.

עטיפת הספריה – האפליקציה

השרת יעבוד בדיוק כמו השרת chat של ביג׳ (selectserver למשל) הלקוח שנמצא במודל ועובד מול ביג יעבוד מולו.
(לצורך כל שאלה – כמו על איזה פורט לרוץ וכדומה – עליכם לעבוד בדיוק כמו השרת של beej)
כאשר הטרד של הריאקטור רץ – הטרד מיין יכול להמתין לו

פורמט ההרצה של השרת הוא:

react_server

(לא צריך ארגומנטים. בדיוק כמו ביג׳)