הערות עבור תרגיל מעשי מס' 2

תרגיל 2 דרש מכם יצירת שרת שבנוי לטפל במספר משתמשים במקביל והתאמת הלקוח שלכם לפיצ'רים נוספים.

הקו המנחה של הבדיקה שלי היה ווידוא נכונות הקוד עפ"י דרישות התרגיל (שימור הפונקציונליות של תרגיל 1 + פונקציונליות הצ'ט) וכן מעבר ידני על הקוד לוודא שלא נוצר פוטנציאל כלשהו להרעבת לקוחות.

במציאות, שרתים שבנויים לטפל בהרבה משתמשים לרוב מקצים תהליך נפרד לריצת כל משתמש והדבר מקטין צורך לסנכרן שליחת הודעות למשתמשים (הצורך לסנכרון קיים, אבל באיזורים אחרים כמו משאבים כלליים). הדרישה לבנות שרת מסונכרן עם select היא דרישה מאתגרת והיו הרבה נקודות עדינות שהיה צריך לשם אליהן במימוש השרת ונתמקד בהן במסמך זה.

קודם כל, שורת הקריאה ל-select היא:

int select(int *n*, **fd_set** **readfds*, **fd_set** **writefds*, **fd_set** **exceptfds*, **struct timeval** **timeout*);

כמעט כולכם הפנמתם את הנושא של שימוש ב-select ע"מ למנוע פעולות קריאה חוסמות מלקוח, ודאגתם לעדכן את הארגומנט השני של select שהוא הreadfds. רבים מכם שכחו שיש ארגומנטים נוספים ל-select ופעמים רבות הניחו שה-socket פתוח באופן קבוע לשליחה, וזוהי שגיאה מבחינת דרישות התרגיל שכן זה יכול ליצור מצב בו אנחנו נחסמים בעת שליחה ללקוח ולמעשה מונעים טיפול בשאר הלקוחות. ארגומנט נוסף ל-select בו כמעט אף אחד לא השתמש (והרבה פעמים הוא מוזנח בדוגמאות) הוא exceptfds. כשאתם מנהלים אצלכם רשימת משתמשים בשרת, פעמים רבות יכול לקרות מצב בו משתמש אחד כשל (יצא באופן לא מסודר) ויש צורך לטפל בשגיאה. שימוש נכון בexceptfds יכול לעזור לכם לנטר רגעים כאלה, ולנקות את כל המשאבים המוקצים עבור המשתמש בשרת. ע"מ לוודא שימור מתן שירות, אחת הבדיקות שהרצתי הכשילה בכוונה משתמש ובדקה את אופן התגובה לשרת. כשמשתמש אחד נכשל, אין שום סיבה שהשרת יפסיק לתת שירות (תארו לעצמכם שכל פעם שהיה נסגר לכם הדפדפן כשאתם באתר, האתר היה קורס).

נקודה נוספת שבלטה אצל אנשים רבים בתרגיל היא שכפול הקוד ושימוש לא נכון בפונקציות הספריה. היו אנשים רבים, שעדיין לא הייתה אצלם בקוד בדיקה לערכי החזרה של ()send ו-recv(). כמו שכתבתי הערות עבור התרגיל המעשי הקודם, זה די בסיסי להחזיק קריאות לפונקציות ספריה בפונקציה מסודרת, ובכך לאפשר ניטור שגיאות ביעילות. שכפול קריאות לפונקציות, מגביר את הסיכוי שבאחת הקריאות לא נבדקו מספיק מקרי קצה, ומצביע על תכנון לא נכון של הקוד.

הקריאה הבודדת לפונקציות ספריה היא חשובה גם בשימוש ב-select. היו אנשים שהחזיקו בקוד 2 קריאות שונות ל-select, אחת עבור כתיבה ואחת עבור קריאה. בהנחה שכל אחת מהקריאות החזיקה timeout, הדבר אינו יכול לגרום להרעבה, אבל עלול לגרום לבעיות אחרות. שימוש מרובה בפונקציות לא חוסמות יכול לגרום להרבה context switches ובזבוז משאבי מעבד כשלמעשה ניתן היה להיחסם על קריאה אחת ל-select גם עבור כתיבה וגם עבור קריאה ולחסוך שורות קוד ובזבוז משאבים.

דוגמה יפה ל-design pattern שיכול לעזור במצבים כאלה היא Reactor והקישור הנ"ל מסביר באופן די טוב על הנושא.