עבודת בית – שרות שיתוף קבצים

את התרגיל יש להגיש עד יום ראשון, ה-5 בינואר 2020 ל-moodle. כללו בהגשה את הקוד, תכנית מקומפלת ודוגמת פלט כמבואר בסוף מאמר זה.

בתרגיל זה נבנה מערכת שיתוף קבצים (peer-to-peer file share או P2P) דוגמת Napster, eMule, Vuze, KaZaA ודומיהם. במערכת כזו כל משתמש נקרא peer, שווה. כל שווה יכול להציע לשתף קובץ הנמצא ברשותו, או לבקש משווה אחר שיעביר לו קובץ שהוצע על ידי האחרון לשיתוף. יש במערכת גם שרת שכל תפקידו הוא לשמש תחנה להחלפת מידע בין השווים.

2. קבצים

המערכת מכילה שלושה קבצי קוד, בהם header אחד ושתי תכניות:

1. p2p.h – קובץ הגדרות המשמש את הקבצים האחרים.
2. p2pserver.c – הקוד של השרת. במערכת חיה פועל עותק אחד של תכנית זו.
3. p2pclient.c – הקוד של השווה. במערכת חיה יכולים לפעול מספר לא מוגבל של עותקים של תכנית זו.

לתרגיל זה מצורף קובץ p2p.h. אבקשכם להשתמש בקובץ זה. אתם רשאים להוסיף עליו הגדרות כראות עיניכם, אבל לא לגרוע מאלה שכבר יש בו.

2. התנהגות המערכת

כלפי חוץ, התכניות מתנהגות באופן הבא:

השרת (p2pserver.c): על השרת להיות מופעל לפני שמריצים תכניות של שווים. התכנית מופעלת ללא פרמטרים ואינה מפיקה כל פלט מיוחד. לצורך הבדיקה אבקש הדפסת הודעות שונות בצמתים חשובים של התכנה על מנת לדווח על ביצועם, אך הודעות אלו אינן מהותיות לפעולה של השרת.

השווה (p2pclient.c): גם תכנית השווה אינה מפיקה פלט פרט להודעות המופקות בצמתים חשובים המציינות שהתבצעו. לעומת זאת, תכנית השווה מופעלת עם פרמטרים בשורת ההפעלה שמציינים מהי הפעולה שהשווה אמור לבצע. האפשרויות הן כדלהלן:

1. seed – השווה מבקש לשתף קבצים הנמצאים בתיקיה הנוכחית בה הוא רץ. לדוגמה, הפקודה הבאה מוסיפה את הקבצים this, that ו-other לאוסף הקבצים שהשווה מבקש לשתף:

```
./p2pclient seed this that other
```

2. leech – השווה מבקש להעביר אליו קבצים שהוכרזו כמשותפים על ידי שווים אחרים. לדוגמה:

```
./p2pclient leech this other
```

כתוצאה מכך, הקבצים this ו-other יועברו על ידי שווים ששיתפו קבצים אלה אל התיקיה הנוכחית שבה השווה המבקש רץ. בעקרון, שני המופעים של השווה שהודגמו כאן יכולים להיות מופעלים על כל שני מחשבים ברחבי האינטרנט.

3. shutdown – השווה גורם לשרת ולכל השווים האחרים שרצים כרגע להסתיים ולשחרר את המשאבים שהם תופסים בטרם יסתיים אף הוא בעצמו. לדוגמה:

```
./p2pclient shutdown
```

נוח להחזיק אופציה כזו על מנת להוריד את המערכת, דבר שבודאי תבקשו לעשות לעיתים תכופות תוך כדי פיתוחה ☺.

3. הפרוטוקול

ההתנהגות המתוארת לעיל מושגת על ידי פרוטוקול שבאמצעותו מתקשרים השווים עם השרת וגם זה עם זה (כמובן זה עם זה – הרי זו מערכת peer-to-peer...). הפרוטוקול מכיל הודעות בעלות מבנה קבוע שמופעי התכניות השונות מעבירים ביניהם דרך רשת התקשורת. הקובץ p2p.h מכיל הגדרות לטיפוסי ההודעות השונים.

טיפוסי ההודעות הם כדלהלן:

1. MSG_NOTIFY – הודעה זו משמשת שווה כדי להודיע לשרת שברצונו להציע קובץ לשיתוף. ההודעה מכילה את שם הקובץ, את כתובת ה-IP של השווה, וגם את מספר ה-port שעליו השווה מאזין לבקשות להורדת קובץ זה. השרת שומר מידע זה. מיד נסביר כיצד נקבע מספר ה-port.
2. ACK_MSG – הודעה זו משמשת את השרת להגיב על הודעת NOTIFY_MSG שקיבל משווה. ההודעה מכילה מספר port, שיוסבר להלן.
3. MSG_DIRREQ - הודעה זו משמשת שווה על מנת לבקש מהשרת את רשימת הקבצים המשותפים.
4. MSG_DIRHDR – השרת מגיב באמצעות הודעה זו להודעת MSG_DIRREQ שקיבל משווה. ההודעה מכילה את מספר הקבצים המשותפים.
5. MSG_DIRENT – בעקבות הודעת MSG_DIRHDR שהשרת שלח לשווה, ממשיך השרת לשלוח הודעות מסוג MSG_DIRENT. כל אחת מכילה שם של קובץ משותף אחד, ואת פרטי אחד השווים שהסכים לשתף אותו: את כתובת ה-IP שלו ואת ה-port שעליו הוא מאזין לבקשות. המידע שהשווה קבל בתוך ההודעה MSG_DIRHDR מאפשרת לו לדעת לכמה הודעות כאלה לצפות.

6. MSG_FILEREQ – הודעה הנשלחת משווה אחד ("המבקש") לשווה אחר ("המספק") במסגרת תקשורת peer-to-peer (שזו הלא מטרת כל העניין). ההודעה מכילה שם של קובץ משותף, אחד מאלה שהשווה "המספק" הודיע לשרת שהוא מוכן לשתף. ההודעה נשלחת לכתובת ה-IP ול-port שקשורים לקובץ המבוקש בהודעה MSG_DIRENT.

7. MSG_FILESRV – תגובת "המספק" לשווה "המבקש" בעקבות הודעת MSG_FILEREQ. ההודעה מכילה את אורך הקובץ המבוקש. בעקבות משלוח הודעה זו, ימשיך "המספק" וישלח את כל תוכן הקובץ אל "המבקש". אורך שלילי מציין שה"מספק" אינו מעוניין להיענות לבקשה.

מקובל ש-"המבקש", אחרי שקבל את הקובץ מ-"המספק", יכריז על עצמו מול השרת כספק נוסף של הקובץ, והופך בכך למספק אפשרי בעצמו עבור קובץ זה.

8. MSG_SHUTDOWN – הודעה שיכולה להשלח לשרת וגם לשווה. כל תכנית שמקבלת הודעה זו אמורה לסגור את הקבצים וה-sockets הפתוחים שלה ולהסתיים.

שווה שולח הודעת MSG_SHUTDOWN אם המלה "shutdown" נמצאת בין הפרמטרים שהתכנית קבלה כשהופעלה. במקרה כזה על השווה לבקש את רשימת הקבצים המשותפים מהשרת, לעבור עליה ולשלוח לכל אחת מהשווים המצויינים בה (באמצעות כתובת IP ו-port) הודעת MSG_SHUTDOWN. לבסוף, עליו לשלוח הודעה כזו גם לשרת, ולהסתיים בעצמו.

כיצד נקבע ה-port:

שווה המציע קובץ לשיתוף **בפעם הראשונה** שולח את שם הקובץ בתוך הודעה מסוג MSG_NOTIFY יחד עם port שמספרו 0. ערך port בלתי חוקי זה מסמן לשרת שזו אכן הפעם הראשונה שלקוח זה מתקשר. השרת אז ממציא ערך חוקי **חדש** ושולח אותו חזרה לשווה בהודעת MSG_ACK. מכאן ולהבא על השווה, שקבל מספר port מהשרת, לשוב ולציין אותו בכל הודעת NOTIFY_MSG שישלח בעתיד, כדי שהשרת לא יתפתה להמציא עבורו מספר חדש.

השרת מקבל בהודעת MSG_NOTIFY, בנוסף לשם הקובץ, גם את כתובת ה-IP של השווה. מספר ה- port נקבע בינו ובין השווה כפי שתואר לעיל. השרת שומר את כל המידע הזה ומספק אותו לכל מי שמבקש אותו (באמצעות MSG_DIRREQ). הטיפוס file_ent_t המוגדר ב-p2p.h נועד למטרה זו. באמצעות מידע זה יודע שווה לאיזו כתובת לפנות על מנת לקבל כל קובץ.

מאידך, שווה שקיבל מספר port מהשרת אמור להאזין על ה-port הזה לבקשות משווים אחרים, ולשרת אותן ע"י משלוח קבצים שהסכים לשתף.

שווה "מספק" רשאי להחליט **להפסיק** לשרת בקשות עבור קובץ מסויים או כל הקבצים שברשותו. במקרה זה, עליו להגיב להודעה מסוג MSG_FILEREQ בהודעת MSG_FILESRV בה שדה האורך הוא שלילי. שווה "מבקש" ינסה במקרה כזה לחפש שווה אחר שהציע את אותו הקובץ ולבקש אותו ממנו.

4. הפלט הנדרש

כאמור, השרת והשווים אינם צריכים להפיק שום פלט על מנת למלא את תפקידם: די שהקבצים שהתבקשו מגיעים ליעדם. אולם כדי לעזור לכם לדבג את התרגיל (ולסייע בידי בבדיקתו), אבקש שכל התכניות תפקנה הודעה לפלט בכל פעם שהן מבצעות אחת מהפעולות הבאות:

1. בכל פעם ששרת פותח socket לשימוש הציבור. על ההדפסה לציין את כתובת ה-IP וה-port של ה-socket.
2. שליחה של הודעה על הרשת או קבלת הודעה מהרשת. יש לציין בהדפסה את סוג ההודעה ומהם ערכי השדות החשובים שבה.
3. כאשר השרת מקצה port חדש לשווה בפעם הראשונה וכאשר שווה מקבל מספר port חדש. על ההדפסה לציין את מספר ה-port.
4. כאשר שווה שולח קובץ וכאשר שווה מקבל קובץ. על ההודעה לכלול את שם הקובץ ואת אורכו. יש להפיק הדפסה מיוחדת במקרה ששווה בחר להתעלם מבקשה של שווה אחר.

על כל ההדפסות להיות בפורמט קבוע, כך שקל לעקוב אחרי ההתפתחויות. על כל הדפסה להתחיל בשורה חדשה ולציין בראשה אם מקורה בשרת (Server), בשווה שמתפקד כלקוח מול השרת (Client), או כשווה מול שווה אחר (Peer). (ראו דוגמא לפלט אפשרי בעמוד הבא.)

לצורך בדיקת המערכת, נזדקק לשלוש תיקיות: א׳, ב׳ ו-ג׳ (או שלושה שמות אחרים, כרצונכם). בכל תיקיה יהיו כמה קבצים, למשל א1, א2 ו-א3 בתיקיה א׳, ב1 ו-ב2 בתיקיה ב׳, וכו׳ (או שמות אחרים). בנוסף, בתיקיה ב׳ יהיה גם העתק של קובץ א1.

אבקש לבצע את הפעולות הבאות:

1. להפעיל את השרת בתיקיה א׳
2. להפעיל שווה בתיקיה א׳ שיציע לשתף את א1 ו-א2.
3. להפעיל שווה בתיקיה ב׳ שיציע לשתף את א1 ו-ב2.
4. להפעיל שווה בתיקיה ג׳ שיבקש לקבל את א2 ו-ב2.
5. למחוק את הקובץ א1 מתיקיה ב׳.
6. להפעיל שווה בתיקיה ג׳ שיבקש לקבל את א1. (יהיה נחמד אם אפשר יהיה לראות שהוא מנסה לקבל את הקובץ מתיקיה ב׳, נכשל ואז מנסה למצוא מקור אחר, למשל מתיקיה א׳.)
7. להפעיל שווה שיסגור את המערכת.

את הפלט שיווצר מפעולות אלו יש להגיש, יחד עם הקוד.

```
$ cc -o server p2pserver.c
$ cc -o client p2pclient.c
$ ./server &
 [2] 3502
Server - server: opening socket on 127.0.0.1:12345
$ ./client seed GoneWithTheWind.mkv EllaInBerlin.flac &
 [2] 3641Server - notify: receiving MSG_NOTIFY
Server - notify: assigned port 12346
Server - notify: sending MSG_ACK
Client - share: sending MSG_NOTIFY for "GoneWithTheWind.mkv" @ 127.0.0.1:0
Peer -   start_server: starting peer server
Peer -   start_server: opened socket
Client - share: sending MSG_NOTIFY for "EllaInBerlin.flac"The @ 127.0.0.1:12346
Peer -   start_server: bound socket to port 12346
Peer -   start_server: listning on socket
Client - share: receiving MSG_ACK
Client - share: receiving MSG_ACK
Client - share: set port to 12346
Server - notify: receiving MSG_NOTIFY
Server - notify: sending MSG_ACK
Server - notify: receiving MSG_NOTIFY
Server - notify: assigned port 12347
Server - notify: sending MSG_ACK
Server - notify: receiving MSG_NOTIFY
Server - notify: sending MSG_ACK
Server - dirreq: Receiving MSG_DIRREQ
Server - dirreq: sending MSG_DIRHDR with count=4
Client - get_list: sending MSG_DIRREQ
Client - get_list: receiving MSG_DIRHDR with 4 items
Client - get_list: received MSG_DIRENT for "ModernTimes.avi" @ 127.0.0.1:12347
Client - get_list: received MSG_DIRENT for "EllaInBerlin.flac" @ 127.0.0.1:12347
Client - get_list: received MSG_DIRENT for "EllaInBerlin.flac" @ 127.0.0.1:12346
Client - get_list: received MSG_DIRENT for "GoneWithTheWind.mkv" @ 127.0.0.1:12346
Client - get_file_from_client: getting file ModernTimes.avi from client on port 12347
Client - get_file_from_client: established connection
Client - get_file_from_client: sent MSG_FILEREQ
Client - get_file_from_client: received MSG_FILESRV: file length 6333
Client - get_file_from_client: opened file <ModernTimes.avi>
Client - get_file_from_client: obtained file ModernTimes.avi from client on port 12347
$ ./client shutdown
Client - get_list: sending MSG_DIRREQ
Client - get_list: receiving MSG_DIRHDR with 4 items
Client - get_list: received MSG_DIRENT for "ModernTimes.avi" @ 127.0.0.1:12347
Client - get_list: received MSG_DIRENT for "EllaInBerlin.flac" @ 127.0.0.1:12347
Client - get_list: received MSG_DIRENT for "EllaInBerlin.flac" @ 127.0.0.1:12346
Client - get_list: received MSG_DIRENT for "GoneWithTheWind.mkv" @ 127.0.0.1:12346
Client - get_list: sending MSG_SHUTDOWN to peer at 127.0.0.1:12346
Client - get_list: sending MSG_SHUTDOWN to peer at 127.0.0.1:12347
Client - get_list: sending MSG_SHUTDOWN to server at 127.0.0.1:12345
```

דוגמה לחלק מפלט אפשרי מהתכניות (זה לא פלט מלא)

בהצלחה!