תרגיל 2

הנחיות הגשה

1. העבודה היא ביחידים.
2. ההגשה היא עד ליום חמישי, בתאריך 8.12.16 בשעה 23:30.
3. הגשת התרגיל תיעשה ע"י submit. עזרה ניתן למצוא באתר:

http://help.cs.biu.ac.il/submit.htm

כדי למנוע בעיות, עדיף להגיש את הקובץ ישירות מחשבון הלינוקס שלכם.

שימו לב: העברת הקובץ דרך Windows יכולה לגרום לכך שלא יעבור קומפילציה. במקרה זה **הציון יהיה 0** ללא זכות לערעור.

4. יש לוודא שהתרגיל מתקמפל ורץ ללא שגיאות על גבי שרת ה-u2
5. בשורה הראשונה(!) של הקובץ אותו אתם מגישים, יש לציין בהערה מס' ת.ז. ושם מלא. לדוג':
   // 123456789 Ofri Keidar
6. בהצלחה ☺

רקע כללי

בתרגול הראשון דיברנו על endianness ועל ייצוג תווים בזיכרון המחשב. הבדל בין מערכות הפעלה שונות (Windows, Unix, Mac) מתבטא גם בשיטת הקידוד של תווים מסוימים. בתרגיל זה נתמקד בהבדל בשמירת קובץ ב- little ו-big endian ובהבדל בקידוד של תו שורה חדשה ('\n'). בתרגיל תידרשו לממש תוכנית שמקבלת שם קובץ קלט ויוצרת קובץ חדש בהתאם ל-flags שיתקבלו.
הערה חשובה – הקפידו לקרוא ולכתוב את הקבצים בתור קבצים בינאריים.

מבנה התוכנית

עליכם להגיש קובץ בשם ex2.c (שמכיל פונקצית main). התוכנית תקבל command line arguments (ארגומנטים ל-main). הארגומנטים שיתקבלו יקבעו את פעולת התוכנית, כלומר איזה קובץ פלט ליצור על סמך קובץ הקלט שיתקבל. הארגומנטים האפשריים הם:

1. שם קובץ קלט ושם קובץ פלט.
2. שם קובץ קלט, שם קובץ פלט, flag המציין את קידוד מערכת ההפעלה בקובץ הקלט ו-flag המציין את קידוד מערכת ההפעלה הרצוי לקובץ הפלט.
3. כמו 2, בתוספת flag המציין האם לשנות את ה-endianness לפיו שמור הקובץ (כלומר, האם להפוך את סדר הבתים).

הערות חשובות

ניתן להניח כי כל תו נשמר בתוך **שני בתים**: המסמכים יישמרו לפי קידוד UTF-16.
יש ליצור קובץ פלט חדש, כך שאם קיים קובץ בשם זה אז יידרס (אבל אפשר להניח שלא יהיה קיים קובץ בשם קובץ הפלט במערכת הבדיקה).
שימו לב- יש לבצע מעבר אחד על קובץ המקור. כלומר, אסור לחזור לתו קודם בקובץ, אלא רק להתקדם לעבר סופו.

כעת נפרט את הנדרש בכל אפשרות.

אפשרות 1- שם קובץ קלט וקובץ פלט ללא flags נוספים

כאשר מתקבלים שמות הקבצים ללא flags נוספים, יש ליצור קובץ פלט שהוא העתק של קובץ הקלט.
הארגומנטים ל-main יתקבלו בסדר הבא:

<source-file-name> <new-file-name>

למשל, הרצת הפקודה הבאה ב-command line:

```
~ ofri$ ./a.out src.txt dst.txt
```

תיצור קובץ בשם dst.txt שהוא זהה בתוכן שלו לקובץ src.txt.

אפשרות 2- הוספת flags של מערכת הפעלה של קובץ הקלט ומערכת הפעלה של קובץ הפלט

ה-flags האפשריים הם:

א. unix– : קידוד תו שורה חדשה בקובץ הוא \n (ייצוג בינארי לפי UTF-16: 0x000a)
ב. mac– : קידוד תו שורה חדשה בקובץ הוא \r (ייצוג בינארי לפי UTF-16: 0x000d)
ג. win– : קידוד תו שורה חדשה בקובץ הוא \r\n . **שימו לב! מדובר בשני תווים צמודים עבור קידוד שורה חדשה** (ייצוג בינארי לפי UTF-16: 0x000d 0x000a)

הארגומנטים ל-main יתקבלו בסדר הבא:

<source-file-name> <new-file-name> <source-file-os-flag> <new-file-os-flag>

נניח ויש לנו קובץ בשם src.txt שנוצר במחשב שמריץ Windows ואנחנו רוצים ליצור קובץ בשם dst.txt שיתאים ל-Unix. כלומר, ליצור קובץ חדש שבו תו שורה חדשה מקודד ע"י \n ולא ע"י \r\n. הרצת הפקודה הבאות ב-command line תיצור את הקובץ dst.txt כנדרש:

```
~ ofri$ ./a.out src.txt dst.txt –win –unix
```

אפשרות 3- בנוסף על אפשרות 2, הוספת flag שקובע האם לשנות את ה-endianness לפיו שמור הקובץ

הקלט יהיה כמו באפשרות 2, בתוספת אחד מה-flags הבאים:

1. swap– : יש להחליף בין שני הבתים של כל תו
2. keep– : אין לשנות את סדר הבתים של כל תו

הארגומנטים ל-main יתקבלו בסדר הבא:

<source-file-name> <new-file-name> <source-file-os-flag> <new-file-os-flag> <byte-order-flag>

למשל, הרצת הפקודה הבא תיצור קובץ כמו באפשרות 2 כאשר שני הבתים של כל תו מוחלפים:

```
~ ofri$ ./a.out src.txt dst.txt –mac –unix –swap
```

בדיקות תקינות הקלט

אם קרה אחד מהמקרים הבאים, אין להדפיס שום דבר ולסיים את התוכנית מבלי ליצור אף קובץ:

1. התקבל רק שם קובץ אחד
2. התקבל רק flag אחד של מערכת הפעלה
3. לא קיים קובץ בשם קובץ הקלט שהתקבל

בדיקת התוכנית שלכם

כדי לבדוק את התוכנית שלכם, תוכלו ליצור קבצי טקסט המקודדים בהתאם לכל אחת מ-3 השיטות.
תוכלו לעשות זאת באמצעות gedit – עורך הטקסט המותקן בלינוקס שלכם.
איך עושים זאת? פותחים קובץ וב-save as בוחרים את הקידוד הרצוי לתו שורה חדשה (Line Ending).
הקפידו לשמור את הקבצים תוך שימוש בקידוד UTF-16 (כך התווים יישמרו על פני שני בתים).

הערות

1. הפונקציות החיצוניות היחידות המותרות בשימוש הן פונקציות המערכת fopen, fread, fwrite, fclose, strlen, strnlen, strcpy, strcmp, strncmp, strstr, sizeof. ניתן גם להשתמש במצביע לקובץ FILE.
חל איסור להשתמש בקוד חיצוני אחר – תוכנית שבה יהיה שימוש בפונקציות או קוד חיצוני אחר תקבל ציון סופי 0.
2. הקלטים לתוכנית יתקבלו בתור command line arguments. הקפידו לקרוא אותם מתוך argv. אין לקרוא קלטים באמצעות scanf.
3. אין להדפיס פלט לתוכנית- התוצר היחיד של התוכנית הוא קובץ הפלט המתאים.

4. הערה חשובה- כדי להשוות בין קבצים (למשל, בין הקובץ שאתם יצרתם לבין קובץ פלט רצוי), השתמשו בפקודה cmp בטרמינל (למשל, cmp file1.txt file2.txt). פקודה זו משווה בין הבתים של שני הקבצים הנתונים.
5. התו "." לא יופיע בשם הקובץ אבל אפשר להניח שהוא תמיד יפריד בין שם הקובץ לבין הסיומת שלו. כלומר, לא יהיה מצב בו שם הקובץ הוא some.file.txt. מצד שני, שמות הקבצים יהיו מהצורה somefile.myextension.
6. ניתן להניח שלא יתקבלו ארגומנטים מיותרים (למשל, שם קובץ שלישי או flag מערכת הפעלה שלישי).
7. בזמן הבדיקה יוכנסו מספר קלטים שונים ויבדקו גם מקרי קצה, אך ניתן להניח כי כל הקלטים שיוכנסו יהיו בהתאם למה שהוגדר בתרגיל. כלומר, ה-flags היחידים שיתקבלו הם אלו שתוארו בתרגיל. בנוסף, ה-flag שמציין את מערכת ההפעלה של קובץ הקלט אכן תואם למערכת ההפעלה עליה נוצר הקובץ.
8. יש להשתמש בכללי התכנות הנכון (הערות, מודולריות, שמות משתנים משמעותיים, הימנעות מ- magic numbers וכו')
9. תוכנית שלא תתקמפל ו/או ,תיכשל בבדיקה תקבל ציון סופי 0 – לא תינתן זכות לערעורים על כך.
10. יש צורך בהערות **משמעותיות** בתחילת הפתרון לכל פונקציה, ורצוי להוסיף הערות גם לקוד עצמו. מתכנת חיצוני (למשל, הבודק של הקורס) שמסתכל על הקוד שלכם צריך להבין בקלות את מהלך התוכנית.
11. הקוד צריך להיות יעיל וקריא.

בהצלחה!