מבוא לתכנות מערכות
תרגיל בית מספר 0

סמסטר אביב 2020

תאריך פרסום : 22/03/2020
תאריך הגשה : 31/03/2020 שעה 23:55
מתרגל אחראי לתרגיל : יורי פלדמן

מענה לשאלות בנוגע לתרגיל יינתן אך ורק בפורום התרגיל במודל. לפני פרסום שאלה אנא בדקו אם כבר נענתה – מומלץ להיעזר בכלי החיפוש שהוצגו במצגת האדמיניסטרציה בתרגול הראשון.

1 הערות כלליות

- **תרגיל זה מהווה 1% מהציון הסופי**

2 הקדמה

מטרת בתרגיל זה היא ביצוע מספר צעדים ראשונים בעבודה מרחוק על שרתים ובסביבת UNIX, כן על מנת להתרגל לסביבת העבודה בקורס. התרגיל מורכב משלושה חלקים :

1. התחברות לשרת csl3 וביצוע פעולות בסיסיות.
2. כתיבת תכנית ראשונה ב-C, הידורה ובדיקתה על שרת ה-csl3.
3. מציאת באגים בתכנית לדוגמה ע״י שימוש בדיבאגר (כלי לניפוי שגיאות).

הערות:

- התרגיל להגשה **ביחידים**.
- יש להגיש את חלק ב׳ וג׳ של התרגיל כך שייבדק על ידי הבודק האוטומטי אשר בשימוש בקורס.

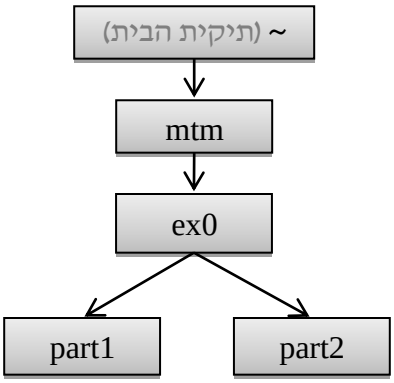
3 חלק א׳ - התחברות ופעולות בסיסיות

1. התחברו לחשבונכם בשרת ה – csl3. הוראות מפורטות על הדרכים השונות לעשות זאת נמצאות במדריך ההתחברות ב-ssh שבאתר תחת Course Material / Guides.
2. לאחר שנפתח בהצלחה חיבור ssh לשרת, ניתן להקליד פקודות, שבלחיצה על enter יתבצעו על השרת, ע״י פקודת ה-shell שהופעלה עם החיבור (בד״כ csh היא ברירת המחדל).
כמה פקודות לדוגמא בהן ניתן להשתמש לצורך התרגיל :

pwd	מדפיסה את המסלול המלא אל התיקייה הנוכחית מה-root (התיקייה ״/״, התיקייה הראשית).
ls <dir>	מציגה את תוכן התיקייה dir (קבצים ותיקיות) אם נקראה ללא פרמטרים, מציגה את תוכן התיקייה הנוכחית.
cd <dir>	מחליפה את התיקייה הנוכחית לתיקייה <dir>
cp <source file(s)> <destination>	מעתיקה את קובץ (קבצי) המקור לקובץ (תיקיית) היעד. ניתן להשתמש בדגל r– (recursive) על מנת להעתיק תיקיות : cp –r source_directory target_directory
rm <file(s)>	מוחקת קבצים. על מנת למחוק תיקיות יש להשתמש בדגל r–.
diff <file1> <file2>	משווה בין שני קבצי טקסט, ומדפיסה למסך את ההבדלים
zip <zipname> <file(s)>	יוצרת קובץ zip עם הקבצים שקיבלה כארגומנטים

pushd <dir>	"דוחפת" את dir למחסנית התיקיות, ומשנה אליה את התיקייה הנוכחית.
popd	חוזרת אל התיקייה שהייתה הנוכחית לפני הקריאה האחרונה ל – pushd (שולפת ממחסנית התיקיות).
man <command>	מציגה תיעוד ל-command ניתן להשתמש בחצים מעלה/מטה לגלילה, להקיש q לחזרה לשורת הפקודה.

3. צרו את התיקיות הבאות תחת תיקית הבית שלכם: תיקיה בשם mtm, בתוכה תיקיה בשם ex0 ובתוכה תיקיות בשמות part1 ו-part2. כך שיווצר המבנה הבא:



4. כעת נרצה להעתיק את הקבצים הדרושים לפתרון שאר חלקי התרגיל מחשבון הקורס לחשבונכם האישי. לשם כך העתיקו את כל הקבצים הנמצאים תחת התיקיה ~mtm/public/1920b/ex0/part1 לתיקיה part1 אותה יצרתם מקודם. ודאו שלאחר ההעתקה אכן הועברו 9 קבצים שונים: test1.in, test1.out, test2.in, test2.out, test3.in, test3.out, test4.in, test4.out, mtm_sol.
5. באותה צורה העתיקו את הקבצים מ-~mtm/public/1920b/ex0/part2. ודאו שאכן הועתקו 5 קבצים שונים: mtm_buggy.c, test1.in, test1.out, test2.in, test2.out.

4 חלק ב' - תכנית ראשונה ב-csl3

בחלק זה נכתוב ונבדוק תכנית המקבלת רשימת מספרים אשר המשתמש מכניס כקלט, וסוכמת את המעריכים של המספרים המהווים חזקה שלמה של 2.

4.1 מפרט התכנית

התכנית אשר תיקרא mtm_tot תופעל משורת הפקודה ותפעל כלהלן:

1. רושמת "Enter size of input:" ומקבלת מהמשתמש כקלט מספר שלם.
2. אם המספר שהתקבל אינו גדול ממש מ-0 התכנית תדפיס "Invalid size" ותסתיים.
3. כעת התכנית מדפיסה "Enter numbers:" ומקבלת מהמשתמש כקלט מספרים שלמים בהתאם למספר שנקבע בשלב 1. אם יש בעיה באחד המספרים בקלט (למשל מוכנסים תווים שאינם ספרות) התכנית מדפיסה "Invalid number" ומסתיימת.
4. התכנית מדפיסה את המספרים המהווים חזקה שלמה של 2 שהוכנסו בקלט ואת סכומם המעריכים בפורמט הבא:
   שורות מהמבנה The number a is a power of 2: a = 2^j לכל חזקה שלמה של 2 שהתקבלה בקלט (כאשר j הוא המעריך), לפי הסדר בו התקבלו בקלט.
   שורה אחרונה במבנה Total exponent sum is b כאשר b הוא סכום המעריכים.

4.2 דגשים והמלצות

- כל הקוד בתרגילי הבית בקורס צריך לציית למוסכמות הקוד (code conventions) המפורסמות באתר תחת Course Material.
- כדי להימנע מבעיות עם הבודק האוטומטי על התכנית להחזיר 0 בכל מקרה.
- בבדיקה האוטומטית, קוד מקבל ניקוד על מקרה בדיקה אם הוא נותן פלט זהה למצופה ומסתיים ללא שגיאות זמן ריצה או שגיאות זיכרון (ובפרט זליגות). הקפידו על הכללים שנלמדו לניהול זיכרון!
  הערה: הקוד שמסופק לכם בחלק הבא אינו מכיל זליגות זיכרון.

4.3 הידור ובדיקה

כדי להדר את התכנית ולהריצה עליכם להשתמש בשורת הפקודה gcc עם הדגלים:
-std=c99 -Wall -pedantic-errors -Werror -DNDEBUG
ודאו ששם קובץ ההרצה הוא אכן mtm_tot.
כלומר:

```
> gcc -std=c99 -Wall -pedantic-errors -Werror -DNDEBUG part1.c -o mtm_tot
```

כדי לבדוק את התכנית מסופקים לכם קבצי בדיקה. הקבצים מכילים קלט לתכנית ופלט צפוי לכל קלט. את קבצי הבדיקה ניתן למצוא תחת תיקית הקורס ב-csl3 בכתובת: ~mtm/public/1920b/ex0/part1
הקבצים test1.in - test4.in הם קבצי הקלט ואילו הקבצים test1.out - test4.out הם קבצי הפלט (בהתאמה). אלו אותם קבצים אשר העתקתם בחלק הקודם של התרגיל.
כדי לבדוק את התכנית בעזרת הקבצים בצעו את הפעולות הבאות:

1. הדרו את הקוד
2. הריצו את התכנית כך שהקלט הסטנדרטי הוא מהקובץ test#.in והפלט הסטנדרטי הוא לקובץ זמני כלשהו. למשל כך:

```
> ./mtm_tot< test1.in > tmpout
```

3. עליכם לוודא שקובץ הפלט הזמני **זהה לגמרי** לקובץ הפלט הצפוי. ניתן לעשות זאת ע"י שימוש בפקודה diff. על מנת ללמוד על הפקודה diff השתמשו בפקודה man כמו שנלמד בתרגול 1. לדוגמה - שימוש ב-diff לבדיקת הקובץ הקודם:

```
> diff test1.out tmpout
```

אם הקבצים זהים לא יודפס כלום, אם יש הבדל יודפסו ההבדלים בין הקבצים.
בנוסף, מסופק לכם קובץ בשם mtm_sol, המהווה גרסה מקומפלת של התכנית אותה אתם צריכים לכתוב. ניתן להשתמש בו כדי לבדוק מקרים נוספים ולייצר טסטים נוספים.

5 חלק ג' - דיבוג

נעבור כעת לתיקיה part2 שיצרנו בחלק א'. תיקיה זו מכילה קובץ קוד בשם mtm_buggy.c וקבצי בדיקה ופלט צפויים.

1. התכנית mtm_buggy אמורה לקלוט מהמשתמש מספר מחרוזות (בדומה לתכנית בחלק הקודם) ולאחר מכן להדפיס את המחרוזת הארוכה ביותר, המחרוזת המינימלית לפי סדר לקסיקוגרפי ואת המחרוזת המקסימלית לפי סדר זה.
   הדרו את התכנית (לא לשכוח את כל הדגלים) ונסו להריץ את התכנית עם קובץ הבדיקה הראשון.
2. התכנית מתרסקת בשגיאת "Segmentation fault", משמעות השגיאה היא שהתכנית מנסה לקרוא ערכים מתאי זיכרון שאינם מוקצים לה. בד"כ שגיאות אלה נובעות ישירות משימוש לא נכון במצביעים או פשוט נסיון לקרוא מצביע שערכו NULL.
   אמנם הקוד בתכנית mtm_buggy.c אינו גדול במיוחד, אך כבר בכמות כזו של קוד יש להשקיע זמן מה למציאת הנקודה בה מתרחשת השגיאה. כדי למצוא את השגיאה הזו בקלות ניתן להשתמש בדיבאגר gdb אשר מותקן על שרת ה-csl3.
   הריצו את הפקודה הבאה אשר מתחילה את הדיבאגר עם התכנית mtm_buggy (כאשר mtm_buggy הוא שם קובץ ההרצה)

```
> gdb mtm_buggy
```

gdb הוא דיבאגר העובד בטרמינל. כדי להשתמש בו יש להכניס פקודות בדומה לשימוש הרגיל בטרמינל. כדי להריץ את התכנית ניתן להשתמש בפקודה run (כדי להפנות קלט ופלט פשוט מוסיפים את ההפניות כמו בד"כ).
נסו להריץ את התכנית עם קובץ הקלט מתוך gdb.
התכנית רצה תחת הדיבאגר כמו בריצה רגילה, אך הפעם כאשר נגיע לגישה הלא חוקית הדיבאגר יעצור את התכנית ויודיע על השגיאה. בשלב זה נוכל להשתמש למשל בפקודה bt כדי להדפיס את מצב מחסנית הקריאות. פקודות נוספות ניתן ללמוד פשוט ע"י שימוש בפקודה help.

3. עדיין קיימת בעיה המקשה עלינו: המידע במחסנית הקריאות אינו מפורט מספיק. כדי לאפשר ל-gdb להדפיס מידע מדויק יותר יש להדר מחדש את התכנית ולהוסיף את הדגל g-. דגל זה שומר מידע עבור דיבאגרים בתכנית ומאפשר להם להתייחס לקוד המקור.
4. צאו מהדיבאגר (ע"י הפקודה quit), הדרו מחדש את התכנית והריצו אותה תחת gdb. הפעם כאשר תדפיסו את מצב המחסנית תקבלו פירוט של השורות בקוד מהן התבצעו הקריאות לכל פונקציה. למעשה, תקבלו את השורה המדויקת בה קרתה השגיאה. שימו לב ששורה זו היא חלק מפונקציה שמימושה לא נתון ולכן עליכם לחפש את הבאג בפונקציות הקוראות לה.
5. תקנו את השגיאה.
6. לאחר תיקון השגיאה הריצו שוב את הקוד ותיווכחו לדעת שקיימת עוד שגיאה בקוד. מצאו ותקנו גם אותה.
7. כעת התכנית עובדת נכון עם הדוגמה הראשונה - אך זה אינו מבטיח את נכונותה. הריצו את התכנית עם קובץ הבדיקה השני ומצאו את השגיאה הנוספת שהוא חושף - לולאה אינסופית. (**הערה:** כדי לעצור תכנית שנתקעה בלולאה אינסופית ניתן ללחוץ על Ctrl+C.)
8. לאחר תיקון כל שלושת השגיאות ודאו שהתכנית מוציאה פלט זהה לזה שבקבצי הפלט.

6 דרישות, הגבלות והערות כלליות

- שימו לב שייצוג תו סוף השורה הוא שונה בין windows (dos) ל-unix (לינוקס, כמו בשרת, או Mac). לכן בהעברת קבצי טקסט (לדוגמא, מקרי בדיקה, או קוד) בין השניים לאחר עריכה, חשוב להריץ בשרת את הפקודה dos2unix או unix2dos אחרי/לפני ההעברה בהתאמה.
- לאחר פתרון התרגיל, אנא הקפידו להריץ את סקריפט בדיקת השפיות שסופק - finalCheck - על zip. ההגשה ממש.

7 הגשה

יש להגיש את חלק ב' וג' בהגשה אלקטרונית. תוכלו לנצל הזדמנות זו להיכרות עם הבודק האוטומטי ולחסוך אי-נעימויות בתרגילים הבאים.

לנוחותכם מסופקת לכם תוכנית "בדיקה עצמית" בשם finalCheck, בתיקיית התרגיל. התוכנית בודקת ש-zip ההגשה בנוי נכון ומריצה את הטסטים שסופקו כפי שיורצו ע"י הבודק האוטומטי. הפעלת התוכנית ע"י:

```
~mtm/public/1920b/ex0/finalCheck <submission>.zip
```

הקפידו להריץ את הבדיקה על קובץ (zip) ההגשה ממש, דהיינו – אם אתם משנים אותו לאחר מכן – הקפידו להריץ את הבדיקה שוב!

את ההגשה האלקטרונית יש לבצע דרך אתר הקורס. תחת Assignments, Exercise 0, Electronic submission.
קובץ ההגשה צריך להיות קובץ zip המכיל שני קבצים: part1.c ו- mtm_buggy.c (מתוקן).

- אין לצרף קבצים שסופקו לכם, על קובץ ה-zip להכיל רק את קבצי ה-C שכתבתם בעצמכם.
- על הקובץ להיות מכווץ כ-zip (לא rar או כל דבר אחר) כאשר קבצי הקוד נמצאים בתיקיה הראשית בקובץ ה-zip.

8 שינויים עדכונים והודעות בנוגע לתרגיל

כל ההודעות הנוגעות בתרגיל ימצאו באתר של הקורס http://webcourse.cs.technion.ac.il/234124 בדף התרגילים. דף זה יכיל שאלות ותשובות נפוצות. רק הודעות דחופות תשלחנה בדואר. עליכם לעקוב אחר האתר והעדכונים שיפורסמו בו.

בהצלחה !