מבוא לתכנות מערכות
תרגיל בית מספר 0

סמסטר אביב 2022

תאריך פרסום : 26/03/2022
זמן הגשה : 10/04/2022 בשעה 23:59
מתרגל אחראי לתרגיל : אלעד קינסברונר

1 הערות כלליות

- **תרגיל זה מהווה 2% מהציון הסופי**
- התרגיל להגשה **ביחידים**.
- מענה לשאלות בנוגע לתרגיל יינתן אך ורק בפורום התרגיל בפיאצה (קישור באתר הקורס) או בסדנות. לפני פרסום שאלה בפורום אנא בדקו אם כבר נענתה – מומלץ להיעזר בכלי החיפוש שהוצגו במצגת האדמיניסטרציה בתרגול הראשון.

2 הקדמה

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

1. התחברות לשרת CSL3 וביצוע פעולות בסיסיות.
2. כתיבת תוכנית ראשונה ב-C, הידורה ובדיקתה על שרת ה-CSL3.
3. מציאת באגים בתוכנית לדוגמה ע״י שימוש בדיבאגר (כלי לניפוי שגיאות).
4. מבוא לעבודה ולניהול גרסאות באמצעות Git.

הערות:

- יש להגיש את חלק ב׳, ג׳ וד׳ של התרגיל כך שייבדקו על ידי הבודק האוטומטי אשר בשימוש בקורס. חשוב להקפיד על שמות הקבצים.
- יש לראות את ההרצאה והסדנה המוקלטים בנושא Git לפני שתתחילו לפתור את התרגיל.

3 חלק א׳ - התחברות ופעולות בסיסיות ב-Unix וב-Git

1. התחברו לחשבונכם בשרת CSL3. הוראות מפורטות על הדרכים השונות לעשות זאת נמצאות במדריך ההתחברות ב-ssh שבאתר תחת Course Material / Guides. **שימו לב – כאשר אתם מתחברים באמצעות SSH ל-CSL3, עליכם להיות על הרשת הטכניונית TechSec (הזמינה, למשל, בספרייה ובחווה) או להיות מחוברים ל-VPN. מדריך להתחברות ל-VPN מצוי בנספח א׳ לתרגיל זה. אם התחברות ל-SSH לא מצליחה, כדאי לבדוק אם ה-VPN התנתק, ואם כן, להתחבר בשנית.**
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 לדוגמה 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
man <command>	מציגה תיעוד ל-command ניתן להשתמש בחצים מעלה/מטה לגלילה, להקיש q לחזרה לשורת הפקודה.

3. צרו תיקייה חדשה בשם "ex0" בתיקיית הבית שלכם, ומשכו את קבצי התרגיל מ-GitHub. קבצי התרגיל מצויים ב-Repository: https://github.com/CS234124/ex0. השתמשו בפקודה **git clone** שהוצגה בהרצאה המוקלטת:

```
> git clone https://github.com/CS234124/ex0.git
```

4 חלק ב' - תוכנית ראשונה ב-CSL3

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

4.1 מפרט התוכנית

התוכנית 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 התוכנית תדפיס "Invalid size" ותסתיים.
3. כעת התוכנית מדפיסה "Enter numbers:" ומקבלת מהמשתמש כקלט מספרים שלמים בהתאם למספר שנקבע בשלב 1. אם יש בעיה באחד המספרים בקלט (למשל מוכנסים תווים שאינם ספרות) התוכנית מדפיסה "Invalid number" ומסתיימת.
4. התוכנית 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

- כדי להימנע מבעיות עם הבודק האוטומטי על התוכנית להחזיר 0 בכל מקרה בפונקציית ה-main שלה.
- בבדיקה האוטומטית, קוד מקבל ניקוד על מקרה בדיקה אם הוא נותן פלט זהה למצופה <u>ומסתיים ללא שגיאות זמן ריצה או שגיאות זיכרון</u> (ובפרט זליגות). הקפידו על הכללים שנלמדו לניהול זיכרון!
  הערה: הקוד שמסופק לכם בחלק הבא <u>אינו</u> מכיל זליגות זיכרון.

4.3 הידור ובדיקה

כדי להדר את התוכנית ולהריצה עליכם להשתמש בשורת הפקודה של CSL3 בתוכנה gcc עם הדגלים:
-std=c99 -Wall -pedantic-errors -Werror -DNDEBUG
ודאו ששם קובץ ההרצה הוא אכן mtm_tot. שימו לב שיש להגיש את כל הקוד שלכם בחלק זה בקובץ part1.c.
כלומר:

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

כדי לבדוק את התוכנית מסופקים לכם קבצי בדיקה. הקבצים מכילים קלט לתוכנית ופלט צפוי לכל קלט. את קבצי הבדיקה ניתן למצוא ב-part1 בתיקייה שהורדתם מ-Git.

הקבצים test1.in - test4.in הם קבצי הקלט ואילו הקבצים test1.out - test4.out הם קבצי הפלט (בהתאמה). אלו אותם קבצים אשר הורדתם בחלק הקודם של התרגיל.
כדי לבדוק את התוכנית בעזרת הקבצים בצעו את הפעולות הבאות:

1. הדרו את הקוד
2. הריצו את התוכנית 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 (שימו לב לרווחים). 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, למשל כך:

```
elad@DESKTOP-02CE1SJ:~$ diff test1.out tmpout
3,5c3,5
< The number 4 is a power of 2: 4 = 2^2
< The number 256 is a power of 2: 256 = 2^8
< Total exponent sum is 10
---
> The number 4 is a power of 2: 4 = 2^1
> The number 256 is a power of 2: 256 = 2^7
> Total exponent sum is 8
```

בנוסף, מסופק לכם קובץ הרצה בשם mtm_sol, המהווה גרסה של התוכנית 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. התוכנית mtm_buggy אמורה לקלוט מהמשתמש מספר מחרוזות (בדומה לתוכנית בחלק הקודם) ולאחר מכן להדפיס את המחרוזת הארוכה ביותר, המחרוזת המינימלית לפי סדר לקסיקוגרפי ואת המחרוזת המקסימלית לפי סדר זה.
הדרו את התוכנית (לא לשכוח את כל הדגלים) ונסו להריץ את התוכנית עם קובץ הבדיקה הראשון.
2. התוכנית מתרסקת בשגיאת "Segmentation fault", משמעות השגיאה היא שהתוכנית מנסה לקרוא ערכים מתאי זיכרון שאינם מוקצים לה. בד"כ שגיאות אלה נובעות ישירות משימוש לא נכון במצביעים או פשוט ניסיון לקרוא מצביע שערכו NULL.
אמנם הקוד בתוכנית 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 השרת CSL3.
הריצו את הפקודה הבאה אשר מתחילה את הדיבאגר עם התוכנית 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 התוכנית ניתן להשתמש בפקודה run (כדי להפנות קלט ופלט פשוט מוסיפים את ההפניות כמו בד"כ). נסו להריץ את התוכנית עם קובץ הקלט מתוך gdb.

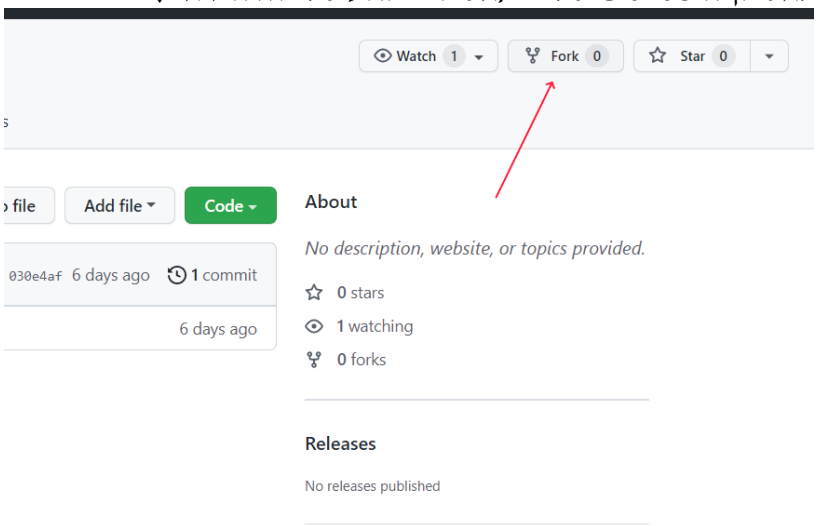
התוכנית רצה תחת הדיבאגר כמו בריצה רגילה, אך הפעם כאשר נגיע לגישה הלא חוקית הדיבאגר יעצור את התוכנית ויודיע על השגיאה. בשלב זה נוכל להשתמש למשל בפקודה bt כדי להדפיס את מצב מחסנית הקריאות. פקודות נוספות ניתן ללמוד פשוט ע"י שימוש בפקודה help.

3. עדיין קיימת בעיה המקשה עלינו: המידע במחסנית הקריאות אינו מפורט מספיק. כדי לאפשר ל-gdb להדפיס מידע מדויק יותר יש להדר מחדש את התוכנית ולהוסיף את הדגל g-. דגל זה שומר מידע עבור דיבאגרים בתוכנית ומאפשר להם להתייחס לקוד המקור.
4. צאו מהדיבאגר (ע"י הפקודה quit), הדרו מחדש את התוכנית 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 התוכנית עובדת נכון עם הדוגמה הראשונה - אך זה אינו מבטיח את נכונותה. הריצו את התוכנית עם קובץ הבדיקה השני ומצאו את השגיאה הנוספת שהוא חושף - לולאה אינסופית. (**הערה:** כדי לעצור תוכנית שנתקעה בלולאה אינסופית ניתן ללחוץ על Ctrl+C.)
8. לאחר תיקון כל שלושת השגיאות ודאו שהתוכנית מוציאה פלט זהה לזה שבקבצי הפלט.

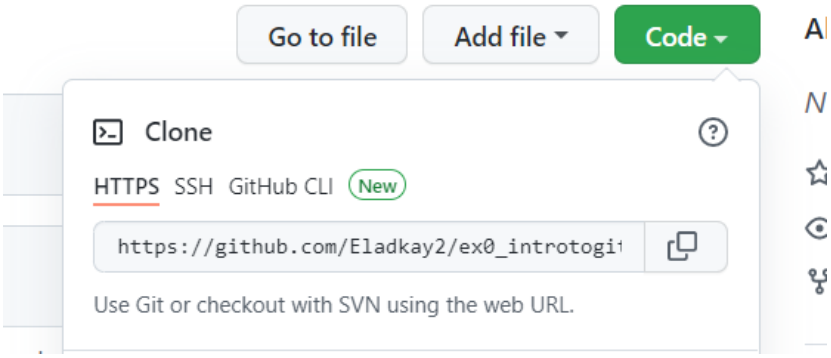
6 חלק ד' – עבודה עם Git

מטרת חלק זה היא היכרות בסיסית עם שיטת העבודה ב-Git.

1. גשו ל-Repository המצוי בכתובת: https://github.com/CS234124/ex0_introtogit ובצעו לו Fork לחשבון ה-GitHub שלכם. (חשבו – מדוע שלב זה הכרחי?)



2. בצעו ל-Repository החדש שיצרתם בפעולת ה-Fork פעולת Clone לחשבונכם ב-CSL3 באמצעות הפקודה שראיתם בהרצאה. **צרפו קישור ל-Repository החדש (שחייב להיות Public) לקובץ טקסט git.txt, וצרפו אותו לתרגיל שלכם:**



3. בתיקייה שתיווצר, תמצאו סקר קצר (צבע אהוב, מאכל אהוב, מוזיקאי/להקה אהובים, סרט אהוב, ציון מבוקש בקורס). מלאו את הסקר כרצונכם. תוכלו לעשות זאת על השרת באמצעות עורכי טקסט טקסטואליים (כמו nano, vim) או לוקאלית על המחשב שלכם עם עורכים גרפיים (כמו Notepad++). במידת הצורך תוכלו להוריד את הקובץ למחשבכם האישי מ-CSL3, ולאחר מכן להחזירו חזרה לשם. **דוגמה** למילוי הסקר:

```
Favorite color: Green
Favorite food: Pizza
Favorite musician/band: Queen
Favorite movie: Back to the Future
Requested grade in the course: 96
```

תמלאו את הסקר עם תשובות לבחירתכם.

4. הוסיפו את הקובץ המתוקן ל-Staging באמצעות הפקודה שראיתם בהרצאה.
5. בצעו לקובץ המתוקן Commit עם הודעת Commit משמעותית שמתארת את מהות השינוי (כפי שדובר בהרצאה).
6. תדחפו (push) את הקובץ המתוקן לremote repository. ודאו ב-Repository שלכם שהשינויים אכן בוצעו גם ב-Remote.
7. עברו ל-Branch הקיים הקרוי bug_fixing באמצעות הפקודה **git checkout**. בצעו פקודת ls ובחנו את המצב החדש של התיקייה.
8. עליכם לתקן את הקוד בקובץ print_number.c כך שהתוכנית תדפיס את מספר הקורס: 234124.
9. בצעו שוב Commit ו-Push לשינוים.
10. בצעו מיזוג בין ה-Branchים main ו-bug_fixing. בצעו בשנית Push וודאו שהשינויים בוצעו גם ב-Remote.

7 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 קובץ zip. ההגשה עצמו.

8 הגשה

יש להגיש את חלקים ב', ג' וד' 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/2122b/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 שלושה קבצים: part1.c, mtm_buggy.c (מתוקן), git.txt.

- 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.

9 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 בדוא"ל. עליכם לעקוב אחר האתר והעדכונים שיפורסמו בו.

בהצלחה !