

מבוא לתוכנות מערכות

תרגיל בית מס' 0

סמסטר אביב 2022

תאריך פרסום : 26/03/2022

זמן הגשה : 23:59 בшу"ב 10/04/2022

מתרגל אחראי לתרגיל : אלעד קינסברונר

1 העrozות כליליות

- תרגיל זה מהו % 2 מהציון הסופי
- התרגיל להגשה **ביחד**.

מענה לשאלות בוגר תרגיל יינתן אך ורק בפורום התרגיל בפייצה (קישור באתר הקורס) או בסדנות. **לפניהם**

פרסום שאלת בפורום אני בדק אם כבר ענתה – מומלץ להיעזר בכל הchiposh שהוצעו במצגת **הADMINISTRATORICA** בתרגול הראשון.

2 הקדמה

מטרת תרגיל זה היא ביצוע מספר צעדים ראשוניים בעבודה מרוחק על שירותי ובסביבת UNIX, על מנת להתרגל לסייע בהובודה בקורס. התרגיל מורכב מארבעה שלבים:

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

העrozות:

- יש להגיש את חלק ב', ג' וד' של התרגיל כך שייבדק 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

התחברו לחשבונכם בשרת CSL3. הוראות מפורטות על הדריכים השונות לעשות זאת נמצאות [במדריך החתחבות ב-ssh](#) שבאתר תחת Course Material / Guides. **שים לב** – כאשר אתם מתחברים באמצעות SSH ל-CSL3, עליהם להיות על הרשות הטכנית TechSec (הזמין, למשל, בספרייה ובחו"ה) או להיות מחובריםVPN. 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 שנפתחה בהצלחה חיבור SSH לשרת, ניתן להקליד פקודות, שבלחיצת על enter יתבצעו על השרת, ע"י פקודת-hell shell שהופעלת עם החיבור (בז"כ csh הוא ברירת המחדל). כמו פקודות לדוגמה בהן ניתן להשתמש לצורך התרגיל:

pwd	מדפסה את המסלול המלא אל התיקייה הנוכחי מה-root (התיקייה //, התיקייה הראשית).
-----	--

ls <dir>	מציגה את תוכן התיקייה dir (קבצים ותיקיות) אם נקראת ללא פרמטרים, מציגה את תוכן התיקייה הנוכחי.
----------	---

cd <dir>	מחליפה את התיקייה הנוכחי לתיקייה <dir>
----------	--

cp <source file(s)> <destination>	מעתקה את קובץ (קובץ) המקור לקובץ (תיקייה) היעד. ניתן להשתמש בדגל -r (recursive) על מנת להעתיק תיקיות:
-----------------------------------	---

<code>cp -r source_directory target_directory</code>	מוחקמת קבצים. על מנת למחוק תיקיות יש להשתמש בדגל <code>r</code> .	<code>rm <file(s)></code>
	משווה בין שני קבצי טקסט, ומדפיסה למסך את הבדלים	<code>diff <file1> <file2></code>
	יצירת קובץ zip עם הקבצים שקיבלה כארוגומנטים	<code>zip <zipname> <file(s)></code>
	מציגה תיעוד ל-command	<code>man <command></code>
	ניתן להשתמש בחצים מעלה/מטה לגיליה, להקשיב לחרורה לשורת הפקודה.	

3. צרו תיקייה חדשה בשם "ex0" בתיקיית הבית שלכם, ומשכו את קבצי התרגיל מ-GitHub. קבצי התרגיל מצויים ב-<https://github.com/CS234124/ex0> : Repository המוקלטת :

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

4 חלק ב' - תוכנית ראשונה ב-CSL3

בחלק זה נכתוב ונבדוק תוכנית המקבלת רשימה מספרים אשר המשמש מכך נסיכון קלט, וסוכמת את המעריכים של המספרים המהווים חזקה שלמה של 2.

4.1 מפרט התוכנית

התוכנית אשר תיראה mtm_tot תופעל משורת הפקודה ותפעל ככללן :

1. רושמת "Enter size of input:" ומתקבל מהמשמש קלט מספר שלם.
 2. אם המספר שהתקבל איינו גדול ממש מ-0 התוכנית תדפיס "Invalid size" ותשטיים.
 3. כתת התוכנית מדפיסה "Enter numbers:" ומתקבל מהמשמש קלט מספרים שלמים בהתאם למספר שנקבע בשלב 1. אם יש בעיה באחד המספרים בקלט (למשל מוכנסים תווים שאינם ספרות) התוכנית מדפיסה "Invalid number" ומשטיים.
 4. התוכנית מדפיסה את המספרים המהווים חזקה שלמה של 2 שהוכנסו בקלט ואת סכום המעריכים בפורמט הבא :
- שורות מהמבנה j The number a is a power of 2: $a = 2^j$ לכל חזקה שלמה של 2 שהתקבל בקלט (כאשר j הוא המעריך), לפי הסדר בו התקבלו בקלט.
- שורה נוספת במבנה Total exponent sum is b כאשר b הוא סכום המעריכים.

4.2 דוגשים והמלצות

- כל הקוד בתרגילי הבית בקורס צריך לצית למוסכמות הקוד (code conventions) [המפורסמות באתר Course Material](#)
- כדי להימנע מבעיות עם הבודק האוטומטי על התוכנית להחזיר 0 בכל מקרה בפונקציית `main` שלה.
- בבדיקה האוטומטית, קוד מקבל ניקוד על מקרה בדיקה אם הוא נותן פלט זהה למינפה ומסתיימים לא שגיאות זמן ריצה או שגיאות זיכרון (ובפרט זליגות). הקפיד על הכללים שנלמדו לניהול זיכרון!
- הערה : הקוד שמספק לכם בחלק הבא אינו מכיל זליגות זיכרון.

4.3 הידור ובדיקה

כדי להדר את התוכנית ולהריצה עליהם להשתמש בשורת הפקודה של CSL3 בתוכנה `gcc` עם הדגמים :

`-std=c99 -Wall -pedantic-errors -Werror -DNDEBUG`

ודאו שם קובץ החרצה 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 לכמ קבצי בדיקה. הקבצים מכילים קלט לתוכנית ופלט צפוי לכל קלט. את קבצי הבדיקה ניתן למצוא ב- part1 בתייה שהורדתם מ-Git.

הקבצים test1.in - test4.in הם קבצי הקלט ואילו הקבצים test1.out - test4.out הם קבצי הפלט (בהתאם). אלו אוטם קבצים אשר הורדתם בחלק הקודם של החומר.

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. עלייכם לוודא שקובץ הפלט הזמני זהה למוגרי לקובץ הפלט הצפוי (שים לב לרוחחים). 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 sol_mtm, המהווה גרסה של התוכנית אותה אתם צריכים לכתוב. ניתן להשתמש בו כדי לבדוק מקרים נוספים וליצור טיסטים נוספים.

5 חלק ג' - דיבוג

נעבור כעת לתקיה part2 שיצרנו בחלק א'. תקיה זו מכילה קובץ קוד בשם mtm_buggy וקובצי בדיקה ופלט צפויים.

1. התוכנית mtm_buggy אמורה לקלוט מהמשתמש מספר מחזורות (בדומה לתוכנית בחלק הקודם) ולאחר מכן להדפיס את המחרוזות הארוכה ביותר, המינימלית לפיקסיקוגרפיה ואת המחרוזות המקסימלית לפי סדר זה.

הדרו את התוכנית (לא לשכוח את כל הדגלים) ונסו להריץ את התוכנית עם קובץ הבדיקה הראשון.

2. התוכנית מתפרקת בשגיאת "Segmentation fault", משמעות השגיאה היא שהתוכנית מנסה לקרוא ערכים מתאי זיכרין שאינם מוקצים לה. בד"כ שגיוט אלה נובעות ישירות שימוש לא נכון במצביים או פשיט ניסיון לקרוא מצביע שערכו NULL.

אמנס הקוד בתוכנית c אין גודל מיוחד, אך כבר בכמה כזו של קוד יש לשקיים זמן מה למציאת הנקודה בה מתרכשת השגיאה. כדי למצוא את השגיאה זו בקלות ניתן להשתמש בדיבאג'r gdb אשר מותן על השרת CSL3.

הריצו את הפקודה הבאה אשר מתחילה את הדיבאג'ר עם התוכנית mtm_buggy (כאשר mtm_buggy הוא שם קובץ ההרצתה)

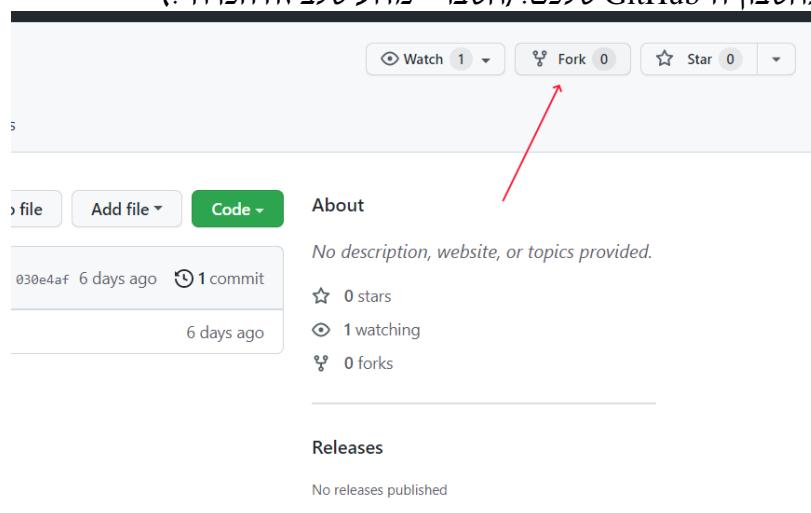
```
> gdb mtm_buggy
```

- gdb הוא דיבאגר העובד בטרמינל. כדי להשתמש בו יש להכנס פקודות בדומה לשימוש הרגיל בטרמינל. כדי להריץ את התוכנית ניתן להשתמש בפקודה `run` (כדי להפנות קלט ופלט פשוט מוסיפים את הפניות כmo בד"כ). נסו להריץ את התוכנית עם קובץ הקלט מתוך `gdb`.
- התוכנית רצה תחת הדיבאגר כמו בritch רגילה, אך הפעם כאשר נגוע לגישה اللا חוקית הדיבאגר יעצור את התוכנית וירodium על השגיאה. בשלב זה נוכל להשתמש למשל בפקודה `bt` כדי להציג את מצב מחסנית הקריואות. פקודות נוספות ניתן למוד פשט ע"י שימוש בפקודה `help`.
3. עדין קיימת בעיה המקשה علينا: המידע במחסנית הكريאות אינו מפורט מספיק. כדי לאפשר ל-`gdb` להציג מידע מדויק יותר יש להדר מחדש את התוכנית ולהוסיף את הדגל `-g`. דגל זה שומר מידע עבור דיבאגרים בתוכנית ומאפשר להם להתייחס לקוד המקורי.
 4. צאו מהדיבאגר (ע"י הפקודה `quit`), הדרו מחדש את התוכנית והריצו אותה תחת `gdb`. הפעם כאשר תציגו את מצב המחסנית תקבלו פירוט של השורות בקוד מהן התבצעו הكريאות לכל פונקציה. למעשה, תקבלו את השורה המדוייקת בה קורתה השגיאה. שימו לב ששורה זו היא חלק מפונקציה שמימושה לא נתנו וכן עליים לחפש את הבאג בפונקציות הקשורות לה.
 5. תקנו את השגיאה.
 6. לאחר תיקון השגיאה הריצו שוב את הקוד ותיווכחו לדעת שקיים עוד שגיאה בקוד. מצאו ותקנו גם אותה.
 7. כתת התוכנית עובדת נכון עם הדוגמה הראשונה - אך זה אינו מבטיח את נכונותה. הריצו את התוכנית עם קובץ בדיקה שני ומצאו את השגיאה הנוספת שהוא חושף - לולאה אינסופית. (**הערה:** כדי לעצור תוכנית שתתקעה לולאה אינסופית ניתן ללחוץ על `Ctrl+C`).
 8. לאחר תיקון כל שלושת השגיאות ודאו שהתוכנית מוציאה פלט זהה לזה שבקבצי הפלט.

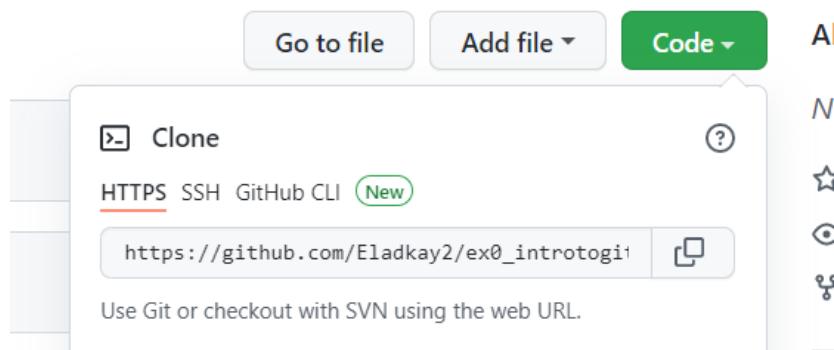
6 חלק ד' – עבודה עם Git

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

1. גשו ל-Repository המצו依 בכתובת: https://github.com/CS234124/ex0_introgit ובצעו לו **fork** לحسابון GitHub שלכם. (חשבו – מדועו שלב זה הכרחי?)



2. בוצעו ל-Repository החדש שיצרתם בפעלת `git clone` למחשבכם ב-CSL3 באמצעות הפקודה שראיתם בהרצאה. **צרפו קישור ל-Repository החדש (שחייב להיות Public) לקובץ טקסט**

git.txt, וצרפו אותו לתרגיל שלכם:

3. בתקייה שתיווצר, תמצאו סקר קצר (צבע אהוב, מאל אל אהוב, מוזיקאי/להקה אהובים, סרט אהוב, ציון מבוקש בקורס). מלאו את הסקר כרצונכם. תוכלו לעשות זאת על השרת באמצעות עורך טקסט טקסטואליים (כמו nano, vim) או לوكאלית על המחשב שלכם עם עורכיים גרפיים (כמו Notepad++). במידת הצורך תוכל להוריד את הקובץ מהמחשב האישי-CSL3, ולאחר מכן להחזירו חזרה לשם.
- דוגמה למילוי הסקר :**

```

1 Favorite color: Green
2 Favorite food: Pizza
3 Favorite musician/band: Queen
4 Favorite movie: Back to the Future
5 Requested grade in the course: 96

```

תملאו את הסקר עם תשובות **לבחירהם**.

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

7 דרישות, הגבלות והערות כלליות

- שיםו לב שייצוגתו סוף השורה הוא שונה בין windows (dos) ל-unix (لينوكس, כמו בשרת, או Mac). לכן בהעברת קבצי טקסט (לדוגמא, מקרי בדיקה, או קוד) בין החניינים לאחר עריכה, חשוב להריץ בשרת את הפקודה dos2unix או unix2dos אשר פניה הינה העברת להתאמה.
- לאחר פתרון התרגיל, אנא הקפידו להריץ את סקריפט בדיקת השפויות שספק - **finalCheck** – על קובץ **.zip**. ההגשה עצמו.

8 הגשה

יש להגיש את חלקים ב', ג' ודי בהגשה אלקטרוני. תוכלו לנצל הזדמנויות זו להיכרות עם הבודק האוטומטי ולהסוך אי-נעימות בתרגילים הבאים.

לנוחותכם מסופקת לכם תוכנית "בדיקה עצמית" בשם **finalCheck**, בתקיית התרגיל. התוכנית בודקת ש-**zip** הוגש בנוי נכון וMRIIZZA את הטסטים שספקו כפי שיורצוו ע"י הבודק האוטומטי. הפעלת התוכנית ע"י:

```
~mtm/public/2122b/ex0/finalCheck <submission>.zip
```

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

את ההגשה האלקטרונית יש לבצע דרך אתר הקורס. תחת heading assignments ,Assignment 0 ,Exercise 0 , יש להעלות קובץ git.txt (מתוקן) וקובץ buggy.c ,mtm_part1.c .
קובץ ההגשה צריך להיות קובץ zip המכיל שלושה קבצים : buggy.c ,mtm_buggy.c ,part1.c .
- אין לצרף קבצים שסופקו לכם, על קובץ ה-zip להכיל רק את קבצי ה-C שכתבתם בעצמכם.
- על הקובץ להיות מכובץ כ-zip (לא rar או כל דבר אחר) כאשר קבצי הקוד נמצאים בתיקייה הראשית בקובץ .zip .

9. שינויים עדכוניים 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 בדואיל. עלייכם לעקוב אחר האתר והעדכוניים שיפורסמו בו.

בהצלחה !