מטלה 1 במערכות הפעלה סמסטר ב' 2025

גירסא 2.0 02/04/2025

נושאים: עבודה ביוניקס. כלים. דיבאג. סיפריות, יצירת תהליכים, signals, pipes, file descriptors.

מטרת התרגיל

- יישור קו בכל הקשור לשימוש בכלים ביוניקס, בניית ספריה, בדיקת הקוד ו-debug, make וכדומה.
- כיסוי קוד, profiling.
- עבודה עם signals ו-PID.
- יצירת תהליכים ועבודה עם pipes.

הנחיות כלליות

- ניתן להגיש את התרגיל על Linux (במכונה וירטואלית), על גבי Mac (יש להתקין Xcode ו-Xcode command line tools) או ב-Windows (יש להתקין WSL).
  תרגילים שיוגשו בסביבת שאינן לינוקס - חובה להשתמש ב-API של POSIX בלבד.
  אסור להשתמש ב-API של COCOA (אפל) או WINDOWS.
- מומלץ לוודא שהקוד רץ גם ב-Linux על מנת לחסוך ערעורים בבדיקה.
- ניתן לכתוב את התרגיל בשפת C או בשפת C++
- נדרש להגיש כל תרגיל בתיקייה נפרדת (קוד וצילומי מסך או קבצים נוספים) נדרש להגיש makefile רקורסיבי לתרגיל שבונה את כל סעיפי התרגיל בעזרת make all ומנקה אותם בעזרת make clean.
- יש להגיש קובץ אחד tar.gz שמכיל את כל סעיפי התרגיל. ניתן לקרוא על tar(1) ועל gzip(1) באינטרנט ובדפי ה-man המתאימים. - ראה הוראות כיצד ליצר tar.gz בסוף.
- יש להגיש את כל תתי המשימות.
- ההגשה בזוגות.
- כל הסטודנטים בקבוצה אחראים על כל התרגיל שהגישו לרבות כל התת משימות ונדרשים להגן על כל התרגיל.

תרגיל 1 - קומפילציה ודיבאג בלינוקס - 10 נקודות

כתבו 3 תוכניות שנופלות באופנים הבאים:

1. גלישה מהמחסנית (לדוגמא עקב רקורסיה אינסופית)
2. חלוקה באפס
3. שימוש בזיכרון לא מוגדר (קריאה או כתיבה מכתובת לא מוגדרת. לדוגמא 0xdeadbeef)

צרו core, פתחו את ה-core בעזרת debugger, הדגימו פתיחה של core עם וללא debug info (כלומר דגל -g בקומפילציה) פתחו את ה-core בעזרת debugger טקסטואלי - הראו איפה הנפילה וערכי משתנים בעזרת פקודת where או print. פתחו את ה-core בעזרת debugger גרפי (לדוגמא ddd) והדגימו את הנפילה בעזרת debugger גרפי. במידה ולא מותקן אצלכם דיבאגר גרפי התקינו אותו (sudo apt install ddd ב-ubuntu.)
הגישו - את הקוד וצילומי מסך של כל השלבים.

תרגיל 2 - שימוש בספריה (נלמד בעזרת ספריית המרוכבים) - 10 נקודות

כתבו תוכנית הבודקת האם מספר מרוכב שייך לקבוצת מנדלברוט (The Mandelbrot set), על התוכנית לקבל 2 ארגומנטים המכילים את חלקי המספר (הממשי והמדומה) וארגומנט אופציונלי N (עם ערך ברירת מחדל לבחירתכם).
שיטת הבדיקה תהיה להלן: חשבו את $a_N(c)$ (לפי ההגדרה בעמוד הוויקיפדיה בעברית), החליטו האם נראה שהסדרה מתבדרת ע"י השוואה של הערך המוחלט שלו (בעזרת carg(3)) למספר גדול כלשהו M (קבוע לבחירתכם). הדפיסו הודעה בהתאם.
דוגמאות:

```
chaim@DESKTOP-RS6Q9F7:~/Ex1$ ./mandelbrot -1.627 -0.001
-1.627 + -0.001i is in the Mandelbrot set
chaim@DESKTOP-RS6Q9F7:~/Ex1$ ./mandelbrot -1.629 -0.001
-1.629 + -0.001i is not in the Mandelbrot set
chaim@DESKTOP-RS6Q9F7:~/Ex1$ ./mandelbrot -1.629 -0.001 5
-1.629 + -0.001i is in the Mandelbrot set
```

- המספר בקבוצה.
- המספר אינו בקבוצה.
- המספר אינו בקבוצה אך סיפקנו N קטן מידי והסדרה לא הספיקה לגדול.

במקרה שהמשתמש לא הכניס מספיק ארגומנטים, יש להדפיס הודעת help פשוטה ל-stderr המתארת את הארגומנטים (בסגנון Usage), לדוגמאות קראו לתוכנה האהובה עליכם עם הפרמטר help–. אין צורך לטפל בארגומנטים שאינם נכונים.

נדרש להגיש makefile, קוד, וצילום מסך של דוגמת הרצה.

תרגיל 3 - בניית ספריה - 10 נקודות

כהמשך לתרגיל 2 בנו את הפונקציה is_in_mandelbrot שחתימתה:

```
bool is_in_mandelbrot(complex double c, int N);
```

הבודקת האם מספר מרוכב שייך לקבוצת מנדלברוט.
קמפלו אותה לספריה דינמית (כלומר shared object) בשם libmandelbrot.so. כתבו קובץ header המכריז על הפונקציה.
כתבו תוכנית המשתמשת בספריה (ע"י include של ה-header ולינקוג' עם ה-shared object) ובודקת שייכות עבור מספרים המתקבלים בקלט מהמשתמש ע"י השורה:

```
scanf("%lf %lf", &real, &imag);
```

ומדפיסה הודעות בהתאם. על התוכנית לעצור כשהמשתמש מכניס את המספר 0+0i (שכמובן בקבוצה).

נדרש להגיש makefile, קוד, וצילום מסך של דוגמת הרצה.

תרגיל 4 - code coverage - משקל 15 נקודות

באתר https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
תוכלו למצוא מימוש (עובד) של אלגוריתם דייקסטרה. (תוכלו לבחור ב-C או C++)

שנו את התוכנית כך שהתוכנית שלכם תתמוך (בתוך לולאת for (;;)) בקבלת גרף חדש, (קריאת הגרף תתבצע מ-stdin בעזרת scanf או cin לבחירתכם), וודאו שקיימות בדיקות תקינות לקלט (כלומר לא שמתי יותר מדי או פחות מדי מרחקים בשורה. דייקסטרה לא תומך במשקלי קשתות שלילים) והרצת האלגוריתם.

בידקו את התוכנית, בדקו את הבדיקה שלכם בעזרת gcov(1).
הראו שהבדיקה שלכם כיסתה את כל הקוד שאתם מגישים כולל מקרי הקצה
(כלומר קלט לא תקין - יותר מדי קשתות בשורה, פחות מדי שורות. קשתות במשקל שלילי וכדומה)

יש להגיש - קוד, make, פלט של gcov(1) צילום מסך של הרצה.

תרגיל 5 - profiling - משקל 15 נקודות

ממשו את שלושת הפתרונות לבעיית max sub array sum בסיבוכיות זמן ריצה של n, n^2, n^3.

התוכניות שלכם יקבלו שני ארגומנטים:
אחד - random seed (לשימוש עם srand)
שני - גודל הקלט (כמות המספרים שהתוכנית תייצר)

הקלט לשלושת האלגוריתמים יחולל באקראי (בעזרת קריאות ל(rand(3) בפונקציה ייעודית והרצת האלגוריתם תתבצע גם היא בפונקציה. המספרים האקראיים יכולים להתפלג בהתפלגות אחידה בקטע (-25, 74).
(אם תרצו התפלגות אחידה אחרת שימו לב שמספרים שליליים חייבים להכלל אחרת התת קטע השלם יהיה התת קטע המלא)

הריצו את שלושת הפתרונות על קלט בגודל 100, 1000, 10000
הדגימו את זמן הרצת האלגוריתם לעומת זמן יצירת המספרים האקראיים בעזרת gprof(1).
נדרש להגיש את הקוד של שלוש התוכניות, make, צילום מסך, פלטים של gprof.

הערה: ניתן למצוא את הגדרת הבעיה ואת שלושת האלגוריתמים בעמודים 3-21 של
https://cses.fi/book/book.pdf - זה גם הספר של קורס תכנות תחרותי (פרסומת סמויה!!)

תרגיל 6 - תקשורת בעזרת סיגנלים - 20 נקודות

בתרגיל זה נכתוב 2 תוכניות שמתקשרות בעזרת סיגנלים - signal_sender, signal_receiver.
על התוכניות להיות מסוגלות לשלוח ולקבל מספר בן 8 ביטים.
התקשורת תעשה בעזרת הסיגנלים SIGUSR1 ו-SIGUSR2, כאשר כל אחד מייצג ערך שהביט יכול לקבל.
השולח יבקש מהמשתמש את ה-PID של המקבל ומספר וישלח את המספר בעזרת הסיגנלים.
המקבל יקלוט את הסיגנלים ובעזרת מטפל/מטפלים ייעודיים ירשום את הביטים ויבנה את המספר (בעזרת שימוש במשתנים גלובליים) ובסוף ידפיס את המספר למשתמש.
דוגמת הרצה:

```
chaim@DESKTOP-2I0PMU2:~/Ex1$ ./signal_sender
Enter receiver PID: 188
Enter message: 42
chaim@DESKTOP-2I0PMU2:~/Ex1$
```

```
chaim@DESKTOP-2I0PMU2:~/Ex1$ ./signal_receiver
My PID is 188
Received 42
chaim@DESKTOP-2I0PMU2:~/Ex1$
```

(התוכנית signal_sender היא היחידה שמקבלת קלט מהמשתמש.)
אסור לתוכניות לתקשר בשום צורה אחרת - קבצים, סוקטים, צינורות, וכו'.
טיפים:

1. וודאו שהסיגנלים לא מתפרצים אחד לטיפול של השני - בעזרת mask.
2. אין תור לסיגנלים - כלומר אם תשלחו מהר מידי הם ילכו לאיבוד, הגבילו את קצב השליחה אך וודאו שהוא לא איטי מידי (כל השליחה צריכה לקחת פחות משנייה אחת).

3. המקבל צריך לחכות לקבלת הסיגנלים, לדוגמה בעזרת sleep, שימו לב מה קורה כשמתקבל סיגנל תוך כדי sleep ומה צריך לעשות אחר כך.

תרגיל 7 - שימוש ב-pipes, יצירת תהליכים - 20 נקודות.

מטרת תרגיל זה - עבודה עם pipe(2), execve(2), fork(2) ולא עבודה עם strings.
תרגיל שיממש את הבעיה בעזרת strings api יפסל.

אני מממש ספר טלפונים בעזרת קובץ טקסט.
בקובץ טקסט נמצאים אצלי כל השמות יחד עם מספר הטלפון במבנה הבא: (שורה לכל שם ומספר, הפרדה בפסיק בין השם למספר, השורה מסתיימת בסימון שורה חדשה)
Nezer Zaidenberg,054-5531415\n

קובץ הטקסט יכול להכיל עשרות ומאות רשומות בפורמט הזה.

לצורך זה אני מממש את 2 התוכניות הבאות -
add2PB - המוסיפה רשומה חדשה לספר הטלפונים (פשוט שורה חדשה).
התוכנית תקבל שם - בדרך כלל שם ושם משפחה - שיכול להכיל רווחים (לדוגמא במקרה של Bat Sheva או שם שני). יכול להכיל רק שם פרטי (אם שם המשפחה לא ידוע לנו או לשמות כמו Mom, Dad) לאחר מכן יהיה פסיק (מובטח לנו ששם לא מכיל פסיק לעולם) ואז מספר הטלפון. סוף רשומה תמיד יהיה line feed.

findPhone - המוצאת את הטלפון של האדם שהתקבל ב argv(1) על ידי קריאה לפקודות
cat(1), grep(1), awk(1), sed(1),cut(1)
יש לייצר תהליכים (בעזרת fork(2) ו-execXX(2) ולהעתיק file descriptor-ים בעזרת dup(2) או dup2(2)
להגיש make וקוד לשתי התוכניות וכן דוגמת הרצה (לדוגמא צילום מסך)
הערה - ניתן לקרוא על grep, awk, sed בכתובת הבאה.
https://www-users.york.ac.uk/~mijp1/teaching/2nd_year_Comp_Lab/guides/grep_awk_sed.pdf
למען הסר ספק - תרגיל זה צריך להיות כתוב ב-C או C++ ולא בbash או PERL.

הקלה חשובה - ניתן להניח שלכל אדם יש רק מספר טלפון אחד.
בנוסף ניתן להניח כי אני מכיר רק אדם אחד בכל שם. אם במקרה ביקשתי שם המופיע בספר הטלפון פעמיים התשובה יכולה להיות כל תשובה שהיא. לדוגמא אם אני מכיר שני אנשים ששמם Avner והקובץ מכיל
Avner Harishon,03-1234567
Avner Hasheni,050-9876543
כל תשובה אפשרית (כולל אף תשובה). בנוסף ניתן להניח שהתו # איננו מופיע כחלק מהשם או המספר באף טלפון.
פתרון אפשרי לבעיית מציאת מספר טלפון בעזרת bash:

```
grep “Micky Mouse” phonebook.txt | sed ‘s/ /#/g’ | sed ‘s/,/ /’ | awk {print$2}
```

הפקודה הראשונה תחזיר רק את השורה שמכילה מיקי מאוס.
הפקודה השניה תהפוך את כל הרווחים לסולמיות.
הפקודה השלישית תייצר רווח במקום פסיק (וכך תיצור עמודה שניה)
הפקודה הרביעית תדפיס את העמודה השניה (כלומר הטלפון)

נספח עבודה עם tar ו gzip

ב 2 פקודות - OLD SCHOOL

tar(1) או tape archive. אספה מספר קבצים (בתיקיה למשל) והדביקה אותם אחד לשני בעזרת הפקודה
tar -cvf mytarfile.tar mydirectory
כדי לפתוח את הקובץ השתמשנו ב
tar -xvf mytarfile.tar

לעיתים קרובות רצינו לדחוס (זיפ) את הקבצים אחד הישומים הפופולריים לדחיסה היה gzip(1) (או gnuzip)
כדי לדחוס השתמשנו ב
gzip myfile
מה שהיה יוצר קובץ דחוס ומוסיף סיומת .gz
כדי לפתוח השתמשנו ב
gunzip myfile.gz

הערה - gnuzip הוא דוחס פופולרי ומהיר אבל קיימים דוחסים אחרים (חלקם דוחסים טוב יותר) למשל
bzip2(1), xzip(1), compress(1)

בפקודה אחת

גירסאות מודרניות של tar(1) יודעות גם לדחוס

tar -zcvf mycompressedfile.tgz mydirectory
ידחוס את התוכן של mydirectory לתוך mycompressedfile.tgz

כדי לפתוח נשתמש בפקודה
tar -zxvf mycompressedfile.tgz

קיימים דגלים אחרים להחלפת הדוחס לפרטים ראה man tar

חשוב

משקל התרגיל 10% מהציון הסופי בקורס.
משקל ההגנה - 5% נוספים. התייחסו לתרגיל בהתאם!