תרגיל 1 במערכות הפעלה סמסטר ב 2023-4

גירסא 1.1 14 לאפריל 2024

נושאים עבודה ביוניקס. כלים. דיבאג. סיפריות, יצירת תהליכים, pipes, file descriptor.

מטרת התרגיל

- ישור קו בכל הקשור לשימוש בכלים ביוניקס, בנית ספריה, בדיקת הקוד וdebug, מייק וכדומה.
- עבודה עם תהליכים בעזרת fork, exec
- יצירת pipe בעזרת pipe והעתקת fd

הנחיות כלליות

- ניתן להגיש את התרגיל על linux (במכונה וירטואלית), על גבי mac (יש להתקין xcode ו xcode command line tools) או בwindows (יש להתקין WSL).
תרגילים שיוגשו בסביבת שאינן לינוקס - חובה להשתמש ב**API** של **POSIX** בלבד.

אסור להשתמש בAPI של COCOA (אפל) או WINDOWS.

- מומלץ לוודא שהקוד רץ גם בLinux על מנת לחסוך ערעורים בבדיקה.
- ניתן לכתוב את התרגיל בשפת C או בשפת C++
- נדרש להגיש כל תרגיל בספריה נפרדת (קוד וצילומי מסך או קבצים נוספים) נדרש להגיש makefie 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 הman המתאימים. - ראה הוראות כיצד ליצר tar.gz בסוף.
- יש להגיש את כל תתי המשימות
- ההגשה בזוגות. סטודנטים שמרגישים שנפגעו עקב המצב (לדוגמא מילואמניקים, בנות זוג של מילואמניקים, מפונים וכדומה) רשאים להצהיר שנפגעו עקב המצב מכל סיבה שהיא ולהגיש בשלשה.
- כל הסטודנטים בקבוצה (יחיד, זוג או שלשה) אחראים על כל התרגיל שהגישו לרבות כל התת משימות ונדרשים להגן על כל התרגיל.

תרגיל 1 - קומפילציה ודיבאג בלינוקס - 10 נקודות

כתוב 3 תוכניות שנופלות באופנים הבאים

1. גלישה מהמחסנית (לדוגמא עקב רקורסיה אינסופית)
2. חלוקה באפס
3. שימוש בזכרון לא מוגדר (קריאה או כתיבה מכתובת לא מוגדרת. לדוגמא xdeadbeef0)

ייצר core, פתח את הcore בעזרת debugger, הדגם פתיחה של core עם וללא debug info (כלומר דגל - g בקומפילציה) פתח את הcore בעזרת debugger טקסטואלי - הדגם איפה הנפילה וערך המשתנים בעזרת פקודת where או print. פתח את הcore בעזרת debugger גרפי (לדוגמא ddd) והדגם את הנפילה בעזרת debugger גרפי. במידה ולא מותקן אצלך דיבאגר גרפי התקן אותו (sudo apt install ddd בubuntu.)

הגישו - את הקוד וצילומי מסך של כל השלבים.

תרגיל 2 - שימוש בספריה (נלמד בעזרת הספריה המתמטית) - 10 נקודות

כתוב את התוכנית *Poisson* המחשבת הסתברות התפלגות פואסנית.
התוכנית תקבל 2 ארגומנטים (בעזרת argc, argv). המייצגים את הערכים λ (ארגומנט ראשון) ו k (ארגומנט שני).
אם התוכנית קיבלה יותר או פחות מ2 ארגומנטים התוכנית תדפיס "Error\n" ותצא.
אם התוכנית קיבלה בדיוק 2 ארגומנטים היא תדפיס את הערך px(k) כלומר

$$p_X(k) = \Pr(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

בדיוק של long double. (אתם נדרשים להשתמש בexpf(3))
שים לב לצורך שימוש בפונקציות של הספריה המתמטית יש לקמפל עם דגל -lm בנוסף יש להוסיף לקוד קריאה לheader של הספריה - בשפת C שם הheader הוא math.h בשפת ++c שם הheader הינו cmath.
נדרש להגיש make וקוד וצילום מסך של דוגמת הרצה

תרגיל 3 - בניית ספריה - 10 נקודות

בעזרת תרגיל 2 בנה את הפונקציה poisson המחשבת התפלגות פואסון.
קמפל אותה לספריה דינמית (כלומר shared object) בשם libpoisson.so
כתוב תוכנית המשתמשת בספריה ומחשבת התפלגות poisson עבור 5 ערכים

ערך	k	λ
1	1	2
2	10	2
3	2	2
4	3	3
5	3	100

ומדפיסה אותם ב 5 שורות (כל ערך בשורה נפרדת)
נדרש להגיש make וקוד וצילום מסך של דוגמת הרצה

תרגיל 4 - code coverage - משקל 20 נקודות

באתר https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
תוכלו למצוא מימוש (עובד) של אלגוריתם דייקסטרה. (תוכלו לבחור בC או C++)
שנו את התוכנית כך שהתוכנית שלכם תתמוך (בתוך לולאת (;;) for בקבלת גרף חדש, (קריאת הגרף תתבצע מstdin בעזרת scanf או cin לבחירתכם) בדיקת תקינות (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).
הראה 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 20 נקודות

ממשו את שלושת הפתרונות לבעיית max sub array sum. (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) 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

תרגיל 6 - שימוש בpipes, יצירת תהליכים - 30 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 ממש את 2 התוכניות הבאות -
add2PB - המוסיפה רשומה חדשה לספר הטלפונים(פשוט שורה חדשה).
התוכנית תקבל שם בדרך כלל - שם ושם משפחה אבל 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) ואז פסיק (מובטח לנו ששם לא מכיל פסיק לעולם) ואז מספר הטלפון. סוף רשומה תמיד יהיה line feed.
findPhone - המוצאת את הטלפון של האדם שהתקבל בargv(1) על ידי קריאה לפקודות
cat(1), grep(1), awk(1), sed(1),cut(1)
יש לייצר תהליכים (בעזרת fork(2) וexecXX(2) ולהעתיק file descriptorים בעזרת dup(2) או 2dup(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 בC 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 פעמים 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
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!