**מכללה אקדמית הדסה**

**החוג למדעי המחשב**

**תרגיל 1#: תכנות מערכת ומבוא לתכנות מקבילי—**

**fork ו- exec**

**תכנית a**

כתבו תכנית בה האב מקבל דרך וקטור הארגומנטים שם של קובץ, וכן ערך טבעי שיש להעביר ל: srand.

האב מגדיר מערך בן מאה אלף תאים.

עתה האב ממלא את המערך כך שהוא יהיה ממוין. הוא עושה זאת ע"י שהוא מכניס לתא #0 ערך אקראי בתחום שבין 0 ל: 10 (כולל 0, ולא כולל 10), ולכל תא נוסף יוכנס הערך שהיה בתא הקודם ועוד ערך אקראי בתחום 0..9.

עתה האב מוליד שני ילדים.

כל ילד. מיליון פעמים, מגריל מספר בין אפס למיליון (כולל 0 ולא כולל מיליון), ומחפש אותו במערך. ילד א' מחפש את הערך חיפוש סדרתי, ילד ב' מחפש חיפוש בינארי.

כל ילד מודד כמה זמן לקח לו להשלים את המשימה.

הילד יפלוט לקובץ (שהאב קיבל דרך וקטור הארגומנטים):

1. את האת s או האות b האם הוא חיפש סדרתית או בינארית
2. כמה ערכים (מתוך מילון הערכים שהוא הגריל) נמצאו במערך,
3. כמה זמן לקח להשלים את המשימה.

הפלט של כל ילד על הקובץ יראה:

s 38177 17

s שכן החיפוש היה סדרתי, 38177 ערכים נמצאו. השלמת השמימה לקחה 17 שניות.

על המשימה הנ"ל האב חוזר עשר פעמים (כלומר עשר פעמים הוא מגריל נתונים למערך, ומוליד שני ילדים, שמחפשים ערכים במערך).

בתום הנ"ל, ברשות האב מצוי קובץ בן עשרים שורות.

האב מחשב ומציג שני מספרים (בשורה אחת, עם רווח ביניהם): כמה זמן, בממוצע רצו עשרת הילדים שביצעו חיפוש סדרתי. כמה זמן בממוצע רצו עשרת הילדים שביצעו חיפוש בינארי.

לבסוף האב מציג בשורה נפרדת נתון נוסף (שלישי): כמה זמן לקח לו עצמו לרוץ מהרגע שהוא החל, ועד שהוא מסיים.

אינכם צריכים להיות מודאגים ממצב בו משך הזמן הנדרש הוא אפס

**התכנית תשכון בקובץ המקור: ex1a.c**

כיצד מודדים זמן ריצה של משימה:

#include <stdio.h>

#include <unistd.h>

#include <sys/time.h>

int main() {

struct timeval t0, t1 ; // there are the necessary variables

gettimeofday(&t0, NULL);

sleep(1) ; // assume this is the task, we want to measure

gettimeofday(&t1, NULL);

printf(" %f\n", (double) (t1.tv\_usec - t0.tv\_usec) / 1000000 +

(double) (t1.tv\_sec - t0.tv\_sec)) ;

return 0 ;

}

הערה: הקפידו מאוד, מאוד, לבדוק האם אין לכם תהליכים (ילדים) שרצים בלולאה אינסופית ולא מסיימים.

1. הפקודה: ps ב- Shell תציג לכם את התהליכים שלכם.
2. היה וגיליתם תהליך כזה, ונניח שמספרו הוא 12345 עצרו אותו ע"י פקודה ה- Shell שהינה:  
    kill 12345

**תכנית b**

1. כתבו תכנית **str\_str** אשר מקבלת באמצעות וקטור הארגומנטים שתי מחרוזות (כלומר ערכו של argc = 3), ובודקת האם המחרוזת הראשונה מופיעה כמות שהיא ברצף בתוך המחרוזת השנייה. במידה וכן יודפס מספר התא במחרוזת השנייה בו מתחילה המחרוזת הראשונה, ובמידה ולא יודפס 1- . שם הקובץ: str\_str.c. (לדוגמה בהרצה: ./str\_str abc xxabcyy יופיע הפלט 2, ובהרצה: str\_str abc xxabxcyy יופיע הפלט 1-). ניתן להניח כי מוזנות שתי מחרוזות.
2. כתבו תכנית **count** המקבלת באמצעות וקטור הארגומנטים שתי מחרוזות, וסופרת כמה מתוויי המחרוזת הראשונה מופיעים במחרוזת השנייה. שם הקובץ: count.c.   
   (לדוגמה בהרצה: ./count xxaaay xy יופיע הפלט: 3)
3. כתבו תכנית **max** המקבלת באמצעות וקטור הארגומנטים מחרוזת הכולל סדרה של מחרוזות, (כלומר ערכו של argc = 2). על התכנית להציג את המחרוזת הגדולה ביותר בקלט, או את המחרוזת error אם לא היה קלט. שם הקובץ: max.c. (לדוגמה: ./max ab abc a תציג את הפלט abc)
4. כתבו תכנית **shell** אשר שוב ושוב קוראת מהמשתמש מהקלט הסטנדרטי פעולה שברצונו לבצע (str\_str, count, max, exit) ואחר את הנתונים לפעולה (כפי שתוארו מעל. הנתונים לכל פעולה יוזנו בשורה יחידה. אפשר להניח זאת, ואין צורך לבדוק זאת). אם שם הפעולה אינו אחד מאלה שצוינו, אזי התכנית מדפיסה: error, וקוראת פעולה מבוקשת חדשה. אם הפעולה היא exit אז התכנית מסיימת, ואחרת התהליך מוליד ילד, הילד מבצע exec לאחת התכניות מעל, תוך שהוא מעביר לתהליך החדש ארגומנטים כנדרש. ההורה ממתין לילדו, ורק אז ניגש לקרוא משמשתמש קוד פעולה חדש. שם הקובץ שמכיל את ה: shell: ex1b.c ה: exec (מתוך shell) יהיה ל: executable ששמותיהם: ./str\_str וכולי.

**הסבר במענה לשאלה שנשאלתי:**

המשתמש יזין ל: shell דרך הקלט הסטנדרטי מחרוזת כגון:

./str\_str abc xxabcyy

כמו תמיד, עת המשתמש מזין קלט למחרוזת הוא לא כותב גרשיים סביב המחרוזת.

ה: shell יקרא את המחרוזת ./str\_str יזהה שזו הפקודה, ולכן עתה יקרא עוד שתי מחרוזות (ואפשר להניח שמוזנות עוד שתי מחרוזות).

עתה ה: shell יוליד ילד.

הילד יירש מאביו את שתי המחרוזות.

הילד יעשה exec ויעביר ל: str\_str בווקטור הארגומנטים (כפי שהיא מצפה לקבל) את המחרוזת.

כלומר ב: str\_str ערכו של argc יהיה בדיוק 3.

בתכנית max המצב מעט שונה: אצלה argc = 2, ולכן כל תת המחרוזות שהיא מקבלת מועברות כמחרוזת אחת שלמה שהיא מפרקת לתת-מחרוזות, ומציגה את הגדולה ביניהן.

**הערה**

ניתן להניח כי כל מחרוזת שנקראת מהקלט ניתן לאחסן במערך בן אלף תאים.

ניתן להניח כי בין כל שתי מחרוזות בקלט מופיע רווח יחיד.

הקלט של המשתמש לכל פקודה יהיה בן שורה אחת: שם הפקודה, ואח"כ הנתונים לה.

בתרגיל זה, באופן חריג, שמות הקבצים לא יהיו: ex1b1.c וכולי, אלא כפי שצוין מעל, וזאת על מנת שהמשתמש יוכל להזין כקלט, למשל, str\_str ואלה יהיו גם שמות קובצי המקור וההרצה המתאימים)

הערות

1. **הקפידו על כל 'כללי הסגנון התכנותי (כולל, למשל, קו מפריד בין פונ', תיעוד משתנים, תיעוד פונ', קבועים,...)**
2. הקפידו על נוהל ההגשה כפי שתואר בתרגיל הקודם.
3. **אל תגידו: לא ידעתי, לא שמעתי, לא הבנתי. זה לא יתקבל בהבנה**.