מערכות הפעלה – המכללה האקדמית תל אביב – יפו

סמסטר ב', תש"פ, 2019-2020

צבי מלמד

תאריך הגשה: מוצ"ש 2/5/2020.

אופן ההגשה: יחידים

תרגיל בית מספר #2

הוראות כלליות

התרגיל הזה מבוסס על תרגיל בית #1. נושאים נוספים שנכללים בו: `fork, exec, redirection`.

הוראות ההגשה המפורטות נמצאות **באתר המעבדה**.

הסביבה הקובעת להרצת התרגילים: סביבת `CYGWIN` במעבדה.

- כלומר... אתם יכולים לפתח ולפתור את התרגילים בסביבה הנוחה לכם. אבל, לפני ההגשה, חובתכם לוודא שהתרגילים רצים בסביבה הפורמלית-סטנדרטית.
- התרגיל מכיל שאלה אחת.

תיאור כללי

בתרגיל זה קיימות שתי תוכניות. התכנית "הראשית" היא `ex2_q1` (וקובץ המקור הראשי שלה הוא `ex2_q1.c`). התכנית הזאת מפעילה תכנית אחרת שנקראת `ex2_q1_helper` (וקובץ המקור שלה הוא `ex2_q1_helper.c`).

הקלט והפלט של התכניות הם כפי שהיו בתרגיל #1, ועליהם מתווספות ההוראות הייחודיות לתרגיל זה.

התכנית "הראשית" (שנקראת כאמור `ex2_q1`) – פועלת בתור "מפצלת". היא קוראת את הקלט (שהגדרתו זהה לקלט בתרגיל #1) ומפצלת אותו לשלושה קבצים, בהתאם לסוג הפוליגון שהקלט עוסק בו:

```
hex_in.tmp – all commands pertaining to new hexagon or writing data about the
nex hexagon, or writing data about all polygongs.
quad_in.tmp – same as above but pertaining to quadrilaterals.
```

לאחר שהתכנית הראשית סיימה לקרוא את הקלט וליצור את שלושת הקבצים הנ"ל, היא מבצעת פעמיים `fork+exec` ומפעילה את התכנית `ex2_q1_helper`. שתי ההפעלות הן: הפעלה אחת עבור משושים, ושנייה עבור מרובעים.

הפלטים של ההפעלות יכתבו לתוך הקבצים `hex_out.log, quad_out.log and`. כלומר, התכנית שקוראת את קובץ הקלט `hex_in.tmp` תכתוב את הפלט שלה לתוך הקובץ `hex_out.log`, וכך בהתאמה לגבי `quad`.

התכנית "המשנית" `ex2_q1_helper` **זהה** לתכנית `ex1_q1` שהייתה בתרגיל #1 פרט להבדל היחיד הבא: הערך המוחזר `n` של התכנית (`exit(n); or return n; from main`) הוא מספר הפוליגונים שהתכנית יצרה. (ניתן להניח שערך זה קטן מ 255 – זה חשוב עבור הפקודה שמשחזרת את הערך המוחזר.

נובע מכך, שהתכנית "הראשית" – `ex2_q1` היא זאת שדואגת ליצירת קבצי הקלט, לביצוע הredirection הדרושים ולהפעלת התכניות `ex2_q1_helper`, בעוד שהתכנית `ex2_q1_helper` איננה יודעת דבר על כך, בדיוק כמו בתרגיל #1.

שני המופעים של התכנית `ex2_q1_helper` צריכים לעבוד במקבילצ (באופן עקרוני, מבחינת כתיבת הקוד. מעשית יכול להיות שאחד יסתיים לפני שהשני יתחיל).

התכנית "הראשית" ממתינה לתכניות העזר שיסתיימו. ברגע שאחת מהן מסתיימת היא כותבת הודעה כזאת לפלט הסטנדרטי:

```
child terminated - created 1 quadrilaterals
child terminated - created 3 hexagons
all child processes terminated
```

שימו לב, כמובן, שהסדר בין שתי ההודעות הראשונות הוא אקראי (ולפיכך יכול להיות שונה).

הנחיות

א. הקפידו שהשמות של הקבצים ושל הארגומנט יהיו בדיוק כפי שמתואר בהוראות למעלה. לשם כך, בצעו העתק-הדבק את הקטע הבא אל התכנית שלכם. אי הקפדה מקשה שלא לצורך על הבדיקה, ולכן תגרור הורדת נקודות!

```
#define HEX_IN    "hex_in.tmp"
#define HEX_OUT   "hex_out.log"
#define QUAD_IN   "quad_in.tmp"
#define QUAD_OUT  "quad_out.log"
```

ב. וודאו שכניסת ה `clean` ב- `Makefile` מוחקת קבצים `*.tmp` וגם `*.log`

ג. הבהרה והדגשה: איזה פקודות בדיוק עוברות לאיזה בן? נניח הבן שעוסק במשושים (ובאותו אופן לגבי הבן שעוסק במרובעים): הוא צריך "לראות" (כלומר קובץ הקלט שלו יכיל) רק פקודות שקשורות למשושים – כלומר, יצירה של משושה חדש או הדפסה של כל המצולעים עד כה (אבל כמובן, שהוא ידפיס רק משושים, כי הקלט שלו לא מכיל מרובעים).

השקיעו מחשבה איזה מלים מהקלט צריך להעביר למי, ואיזה סינון צריך לבצע.
לדוגמא: אם יש לנו פקודה שבה יש פוליגון חדש, מסוג (למשל) משושה, וההדפסה היא של כל המצולעים, – התהליך שמטפל במשושים יקבל מן הסתם את הפקודה (ומיד לאחריה את נתוני הקודקודים). ייצור את המשושה ויבצע את ההדפסה. אותה פקודה תעבור גם לתהליך שמטפל במרובעים, אבל שם, היא רק תגרום להדפסה, ולא ליצירת מרובע חדש. כמו כן, בקובץ הקלט של המרובעים, לא יופיעו נתוני הקודקודים.

ד. יש להגיש פתרון "נקי". למשל, יהיו דיי הרבה קטעי קוד שכתבתם בתרגיל הקודם, וכעת הם מיותרים בתכנית `ex2_q1.c` (בעוד שהתכנית `ex2_q1_helper.c` תישאר כמעט ללא שינוי). הציפייה היא שהתכנית הזאת תכיל רק את מה שהיא צריכה להכיל.

טיפים

להלן כמה טיפים שמבוססים על בעיות (באגים) של תלמידים וכן על הניסיון שלי במימוש התרגיל.

א. באג פשוט אבל קצת קשה לאיתור: בתכנית הראשית (ראשונה) אתם פותחים את הקובץ `hex_in.tmp` וכותבים אליו. בסיום העבודה, המכוון נמצא בסוף הקובץ. עליכם לסגור ולפתוח אותו מחדש לפני ה `EXEC`, או, פשוט יותר להחזיר את המכוון להתחלה ע"י `LSEEK`.

ב. לגבי סוף הקלט – יתכן שהפקודה האחרונה, שבה דלוק הביט שמסמן את סוף הקלט, נכתבת רק לאחד מהקבצים, ולא לאחרים. הפתרון הכי פשוט – הוסיפו פקודה שרק מציינת את סוף הקלט (המספר 1) לכ"א מהקבצים בסופו.

ג. נניח שקלטתם מספר מהקלט – למשל המילה הבאה (בת 64 סיביות) שמתארת קודקודים:

```
0000f605000a0a0a
```

כאשר תכתבו אותו לקובץ (למשל `hex_in.tmp`) סביר שהיא תיכתב בצורה הבאה:

```
f605000a0a0a
```

זה תקין לגמרי! אם לא הורינו ב-`format` של הפונקציה `printf` לרפד באפסים מובילים (אין צורך בכך), זה מה שייכתב, וזה תקין לגמרי.

ד. בביצוע `redirection` יתכן שתמצאו שנוח יותר להשתמש בפונקציה `dup2`.

ה. בהנחה שפתחתם את הקבצים ע"י `OPEN` – אזי קיבלתם `file-descriptor`. נוח להשתמש בפונקציה `dprintf`. (בכדי להימנע מאזהרה ייתכן שתצטרכו להשתמש בדגל `-std=gnu11`).

ו. כל הפונקציות שהזכרתי כאן, ואינן מוכרות לכם – השתמש בפקודה `man` (מתוך הטרמינל) – למשל:
`man dup2`

ז. בכדי שה `make` ירוץ גם כאשר קוד/ערך היציאה מתכנית שונה מאפס, הריצו אותו עם הדגל `-k` (האות `k` זה כנראה מהמילה `keep-on-going`). [במידה וזה נחוץ. זה עשוי להועיל לכם בשלבי הפיתוח]

בהצלחה!!