מערכות הפעלה – המכללה האקדמית תל אביב – יפו

סמסטר ב', תש"פ, 2019-2020
צבי מלמד

גרסה 1.1 . עדכון אחרון: 18/5/20

תאריך הגשה: מוצ"ש 23/5/2020.

אופן ההגשה: יחידים

עדכונים:

1. תוספת 18.5: שימו לב שאין בקוד שנתתי לכם **אין** קריאה ל `sem_wait`. עליכם לשלב אותה בקוד שאתם כותבים. (סעיף ב' בהנחיות).

תרגיל בית מספר #3

הוראות כלליות

הוראות ההגשה המפורטות נמצאות **באתר המעבדה** (אותן הוראות כמו לתרגיל בית #1).

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.

קובץ ה `MAKEFILE` שאתם מגישים, צריך לייצר את כל מה שנדרש לתרגיל (מספר תכניות). כמו כן הוא צריך להכיל כניסת `clean` וגם כניסת `test`. כניסת ה `test` צריכה להכיל בדיוק, את ההפעלה (השורה) הבאה:

```
<tab>./ex3_q1  1000 2000
```

לתרגיל הזה מצורפים הקבצים `ex3_q1_given.h and ex3_q1_given.c`. עליכם להשתמש בקבצים האלו, כלומר לבצע `include` לקובץ `h`, וה- `Makefile` צריך לקמפל את הקובץ `ex3_q1_given.c`. אין (כלומר "אסור") לשנות את הקבצים האלו! (אתם יכולים לשנות ערכים בהם בזמן הפיתוח. התכנית שלכם חייבת לעבוד עם הקבצים המקוריים, כי הם אלו שישתמשו בהם בבדיקה).

תיאור כללי

בתרגיל זה עליכם לכתוב שלוש תכניות. שתיים מאוד קצרות ובסיסיות (ודומות זו לזו). התכנית השלישית היא מהות התרגיל. תכנית זאת - התכנית "הראשית" היא `ex3_q1` (וקובץ המקור הראשי שלה הוא `ex3_q1.c`). מייצרת פייפים ותהליכים בנים, שחלקם מפעילים את התכניות "הקצרות".

התכניות "הקצרות" – cousin_prime.c and twin_prime.c

זוג מספרים ראשוניים נקראים `twin primes` אם יש ביניהם הפרש של 2. לדוגמא, 11, 13 או 101, 103.

זוג מספרים ראשוניים נקראים cousin primes אם יש ביניהם הפרש של 4. לדוגמא, 7, 11 או 13, 17 או 19, 23.

עליכם לכתוב תכנית שקוראת מספרים (בייצוג בינארי, לא הקסא!) כל עוד אפשר, מהקלט הסטנדרטי. היא בודקת אם מספר שהיא קוראת הוא אחד מתוך מזוג של TWIN או COUSIN. אם כן – היא מדפיסה הודעה **ומסתיימת**.

כמו כן, התכנית מסתיימת ברגע שכבר לא ניתן לקרוא מספרים מהקלט.

הנחיות:

- אתרו את הפונקציות הרלבנטיות לזיהוי ולהדפסה בקובץ הנתון, והשתמשו בהן.

התכנית הראשית – ex3_q1.c

בתכנית זאת נדבר על סוגי התהליכים הבאים:

(א) התהליך הראשי, שמגריל מספרים אקראיים.
(ב) תהליכים מסננים, שמסננים את המספרים האקראיים, ומעבירים הלאה רק את המספרים הראשוניים מתוכם.
(ג) תהליכים "תאומים" או "בני-דודים" שאחראים על מציאת המספרים בעלי התכונות האלו.

מוגדרים כבר הקבועים הבאים (בקובץ ex3_q1_given.h)

```
#define PRIME_CHECKERS <value>
#define TWIN_COUNT     <value>
#define COUSIN_COUNT   <value>
```

המטרה בתכנית היא להגריל מספרים בתחום מסוים, לאתר ולהדפיס זוגות מספרים שהם twin-primes או cousin-primes. באופן ספציפי, רוצים להדפיס TWIN_COUNT זוגות של מספרים ראשוניים תאומים, ו-COUSIN_COUNT זוגות של מספרים ראשוניים בני-דודים.

התהליך הראשי מגריל מספרים אקראיים, עד שהוא "מבין" שכבר לא צריך, כלומר יוצרו כל זוגות התאומים והבני-דודים הדרושים.

נניח, שהחלק "הכבד" בתכנית הוא זיהוי מספרים ראשוניים. ולכן, נרצה PRIME_CHECKERS תהליכים שירוצו במקביל, ויסננו את המספרים האקראיים שהתהליך הראשי הגריל. המספרים הראשוניים (אלו שעברו את הסינון) עוברים הלאה לשני תהליכים אחרים, שבודקים אם כל מספר ראשוני שהועבר, הוא תאום או בן-דוד.

תיאור מפורט של מהלך התכנית:

הארגומנטים להפעלת התכנית הם שני מספרים hi and lo שמציינים את תחום המספרים להגרלה.

תחילה התהליך הראשי מייצר PRIME_CHECKERS תהליכים מסננים. תהליכים אלו מסננים את המספרים הראשוניים שהוא יגריל.

לאחר מכן, הוא מייצר שני תהליכים – תאומים ובני-דודים. הראשון דואג לכך שייווצרו ויודפסו למסך TWIN_COUNT זוגות של מספרים ראשוניים תאומים. השני דואג לכך שייווצרו ויודפסו COUSIN_COUNT זוגות של מספרים ראשוניים בני-דודים.

לאחר מכן, התהליך הראשי מגריל מספרים אקראיים, וכותב אותם לפייפ, שממנו קוראים התהליכים המסננים, עד שהוא "יודע" שכבר לא צריך, כלומר יוצרו כל זוגות התאומים והבני-דודים הדרושים ואז הוא מפסיק להגריל. (איך הוא "יודע" – הסבר בהמשך).

התהליך הראשי ממתין שכל התהליכים בנים שלו יסתיימו, ואז הוא מדפיס את ההודעה הבאה ומסתיים:

```
all child processes terminated. Exiting
```

התהליכים המסננים, קוראים מספרים אקראיים מהפייפ. הם בודקים אם זה מספר ראשוני, ואם כן, מעבירים את המספר הזה לשני התהליכים שאחראים לבדיקת "תאומות" או "בן-דודות". כלומר, אותו מספר נכתב לשני פייפים שונים. הם כותבים את המספר לפייפ של ה TWIN או של ה COUSIN רק אם צריך. ברגע שלא צריך לכתוב לאף אחד מהפייפים, ("אין להם כבר מה לעשות") – הם מסתיימים.

התהליכים תאומים או בני-דודים, אחראיים כאמור להדפסת COUSIN_COUNT or TWIN_COUNT זוגות של מספרים לפלט. לשם כך, הם מפעילים את התכניות twin_prim או cousin_prime. שימו לב, שכל הפעלה של התכניות האלו טובה רק ליצירת זוג אחד של מספרים (כך הן כתובות – ראה בסעיף שמתאר את התנהגותן).

להלן דוגמת הרצה של התכנית:

בהנחה שהוגדרו:

```
#define TWIN_COUNT      2
#define COUSIN_COUNT    6
```

```
==> ./ex3_q1 1000 2000
successul unlink of sem_twin.going to open with count=2
successul unlink of sem_cousin.going to open with count=6
TWIN Primes: 1667, 1669
COUSIN Primes: 1483, 1487
TWIN Primes: 1721, 1723
COUSIN Primes: 1873, 1877
COUSIN Primes: 1279, 1283
COUSIN Primes: 1609, 1613
COUSIN Primes: 1999, 2003
COUSIN Primes: 1303, 1307
All child processes terminated. Exiting

tzvi 03:30 PM ~/os_2020b/home_ex/tzvi_solution/012345678/
==>
```

כיצד התהליך הראשי או המסננים יודעים מתי להפסיק?

התהליך הראשי או המסננים צריכים לדעת שנוצרו הכמות הדרושה של זוגות מכל סוג, ולפיכך להפסיק לכתוב לפייפ הרלבנטי.

המידע הזה צריך לעבור בין תהליכים, ונשתמש בסמפורים למטרה הזאת.

לרשות התהליכים האלו עומדות הפונקציות is_twin_done ו- is_cousin_done. עיינו בקוד ושימו לב כיצד הן עובדות. בכדי שמנגנון הסמפורים יעבוד, צריך לייצר אותם, ולאתחל אותם לערך הנכון. בשביל זה, התהליך הראשי, ממש בתחילת ריצתו, יקרא לפונקציה open_all_sem. עיינו גם בקוד הזה.

הנחיות וטיפים:

א. בהתאם לתגובות התלמידים, יתכנו עדכונים או תוספות למסמך הזה. בדקו מדי פעם אם משהו התחדש. (אל דאגה - לא יהיו שינויים משמעותיים).

ב. עדיין לא למדנו על סמפורים, ולכן נתתי לכם את הפונקציות שנתתי – אתם מוזמנים לעשות בעצמכם את המעבדה על סמפורים (התכנית שלי, לעשות אותה בשבוע הקרוב). תוספת 18.5: שימו לב שאין בקוד שנתתי לכם אין קריאה ל sem_wait. עליכם לשלב קריאה כזאת בקוד שאתם כותבים.

ג.

ד. בתרגיל צריכים שלושה פייפים. מומלץ לתת להם שמות משמעותיים (יותר מאשר pp1, pp2, . .).

ה. אם כותבים לפייפ שכל קצות הקריאה שלו נסגרו, התכנית תקבל סיגנל שנקרא SIGPIPE ותסתיים על המקום. בכדי למנוע את זה עליכם לקרוא לפונקציה signal:

```
signal(SIGPIPE, SIG_IGN);  // ignore SIGPIPE signal
```

ו. אפשרות נוספת, ואולי עדיפה, להשתמש בפונקציה שנקראת טפלית סיגנלים, או signal-handler. אתם מודיעים לקרנל, שכאשר קורה הסיגנל הזה, הוא צריך לבצע קריאה לפונקציה הזאת. כך אתם יודעים שהתכנית שלכם קיבלה את הסיגנל הזה. למשל הקוד הבא:

```
signal(SIGPIPE, sigpipe_handler); // notify kernel about my signal handler

// --- and this is the function, i.e. the handler
void sigpipe_handler(int s)
{
  fprintf(stderr, "*** %s *** got signal sigpipe s=%d\n", __func__, s);
}
```

ז. השתדלו לכתוב קוד קומפקטי. לדוגמא, שימו לב שהטיפול ב TWIN או ב COUSIN הוא כמעט זהה.

ח. עבדו בזהירות. תכניות עם פייפים יכולות לשרוף זמן אם משהו לא עובד. התחילו בכך שתממשו רק את החלק של האבא. אח"כ הוסיפו את התהליכים המסננים (התחילו עם אחד). אח"כ הוסיפו את TWIN ואת COUSIN (אולי תרצו להוסיף קודם אחד מהם, ורק אח"כ את השני).

ט. כיתבו תכנית עזר שמייצרת מספרים וכותבת אותם (בינאריים) לפלט הסטנדרטי. וזאת בכדי שתוכלו לבדוק בנפרד את שתי התכניות הקצרות.

בהצלחה!!