Homework 4 Dry

אריאל ווייס 204361315

איליה רוזנצוייג 320517550

יש לנמק כל תשובה, תשובות ללא נימוק לא יתקבלו.

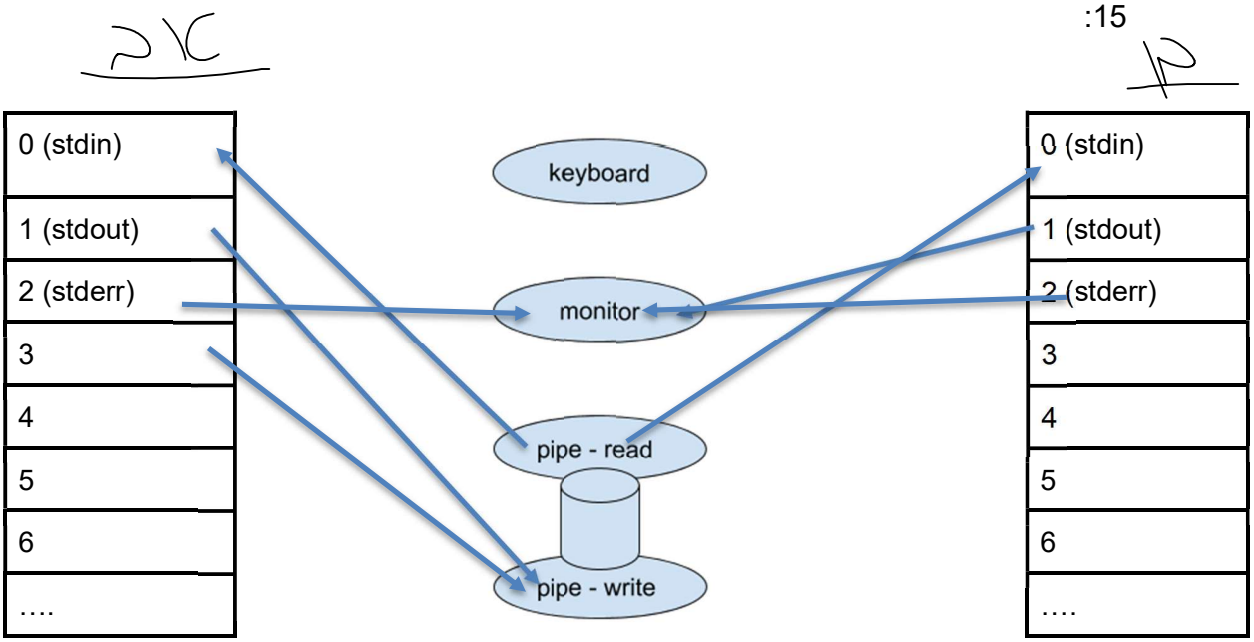
Question 1 - Pipes & I/O:

נתון קטע הקוד הבא:

```
void transfer() {// transfer chars from STDIN to STDOUT
    char c;
    ssize_t ret = 1;
    while ((read(0, &c, 1) > 0) && ret > 0)
        ret = write(1, &c, 1);
    exit(0);
}

int main() {
    int my_pipe[2];
    close(0);
    printf("Hi");
    pipe(my_pipe);
    if (fork() == 0) { // son process
        close(my_pipe[1]);
        transfer();
    }
    close(1);
    dup(my_pipe[1]);
    printf("Bye");
    return 0;
}
```

1. השלימו באמצעות חצים את כל ההצבעות החסרות באיור הבא (למשל חץ מ- stdin ל- keyboard), בהינתן שתהליך האב סיים לבצע את שורה 19 ותהליך הבן סיים לבצע את שורה 15:



2. מה יודפס למסך בסיום ריצת שני התהליכים? (הניחו שקריאות המערכת אינן נכשלות):

a. Hi
b. Bye
c. HiBye
d. לא יודפס כלום
e. התהליך לא יסתיים לעולם
f. לא ניתן לדעת, תלוי בתזמון של התהליכים

נימוק:
הHi יודפס שכן אין כל מניעה לכך. לאחר מכן, במידה שהבן ירוץ קודם, לא יהיה תוכן בpipe שכן האבא עוד לא ביצע את הכתיבה אליו, לכן הוא ייחסם עד שהאבא יכניס את התוכן ולאחריו הבן יוכל להדפיס אותו. אחרת, האבא יכניס את המידע לpipe והבן שינסה להדפיס את המידע הזה למסך יצליח. ובשני המקרים יודפס בסוף HiBye

בסעיפים הבאים נתבונן בקטע קוד חדש, המשתמש בפונקציה transfer מהסעיף הקודם:

```
int my_pipe[2][2];
void plumber(int fd) {
     close(fd);
     dup(my_pipe[1][fd]);
     close(my_pipe[1][0]);
     close(my_pipe[1][1]);
     transfer();
}

int main() {
     close(0);
     printf("Hi");
     close(1);
     pipe(my_pipe[0]);
     pipe(my_pipe[1]);

     if (fork() == 0) { // son 1
          plumber(1);
     }
     if (fork() == 0) { // son 2
     plumber(0);
     }
     printf("Bye");
     return 0;
}
```

3. מה יודפס למסך כאשר תהליך האב יסיים לרוץ? (הניחו שקריאות המערכת אינן נכשלות) רמז: שרטטו דיאגרמה של טבלאות הקבצים כפי שראיתם בסעיף 1.

a. Hi
b. Bye
c. HiBye
d. ByeHi
e. לא יודפס כלום
f. לא ניתן לדעת, תלוי בתזמון של התהליכים...

נימוק:
מכיוון שכבר בהתחלה, עוד טרם ה)fork)ים אנחנו סוגרים את הstdout, דרכו אנחנו מדפיסים למסך, כל תהליך ישכפל את הDiscriptor Table של האב ללא התקן זה. לפיכך, אף אחד מן הבנים לא יוכל לבצע transfer למסך שכן stdout לא יהיה בDT שלהם. בשורה התחתונה – אנחנו נדפיס את Hi בלבד. כמו כן, הבנים מעבירים את הקלט בין הpipים שלהם במעגליות.

סנטה קלאוס שמע שסטודנטים רבים בקורס עבדו במהלך הכריסמס על תרגיל הבית, ואפילו נהנו ממנו יותר מאשר במסיבת הסילבסטר של הטכניון. בתגובה נזעמת, סנטה התחבר לשרת הפקולטה והריץ את התוכנית הנ"ל N פעמים באופן סדרתי (דוגמה ב-bash, כאשר out.a הוא קובץ ההרצה של התכנית הנ"ל):

>> for i in {1..N}; do ./a.out;

4. אחרי שהלולאה הסתיימה, נשארו במערכת 0 או יותר תהליכים חדשים.
מה המספר המינימלי של סיגנלים שצריך לשלוח באמצעות kill על מנת להרוג את כל התהליכים החדשים שסנטה יצר?

a. 0
b. 1
c. N
d. N/2
e. 2N
f. לא ניתן לדעת, תלוי בתזמון של התהליכים...

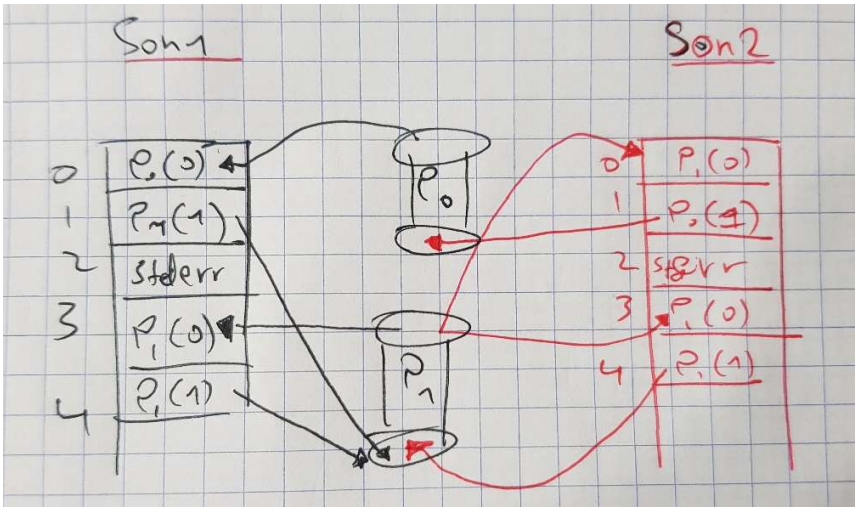
נימוק:
בשאלה הקודמת ראינו שיש מעגל אינסופי בין הpipים של שני הבנים. לפיכך, עבור הרצת N התוכניות, נקבל N זוגות תהליכים שתלויים אחד בשני ויוצרים N מעגלים אינסופיים. לפיכך, מספיק להרוג את אחד הבנים שנמצאים במעגל אינסופי (לכל N הזוגות) כדי להרוג את כל התהליכים.
נשים לב כי ייתכנו שני מצבים כאשר נהרוג את אחד הבנים: אם בPIPE יש מידע שאותו נרצה לכתוב לבן שהרגנו, אזי נקבל סינגל SIGPIPE . אחרת, אם אין מידע בPIPE, הבן שלא נהרג ינסה לקרוא מידע ממנו, יראה כי אין כותבים ולכן יבצע exit(0).

5. מה תהיה התשובה עבור הסעיף הקודם אם נסיר את שורות 5-6 מהקוד?

a. 0
b. 1
c. N
d. N/2
e. 2N
f. לא ניתן לדעת, תלוי בתזמון של התהליכים...

נימוק:

נשרטט את שני הבנים ונראה כי **רק** הריגת הבן השני תוביל לסיום מוצלח של כל התהליכים:



במצב בו בן 1 הסתיים לרוץ ויש בP1 נתונים, בן2 יקרא אותם אבל P1 לא יסגר שכן עדיין יש מצביעים לכתיבה בו, אותם המצביעים שכעת לא סגרנו בשורות 5-6, כלומר בן2 ייחסם לנצח. אנחנו מוכרחים להרוג את בן 2 אם כך.

נראה כי סגירת בן 2 מספיקה:

במקרה שבו P0 מכיל תוכן – בן 1 יקרא אותו, יכתוב לP1 ויצא, שכן אין יותר מצביעים לכתיבה (סגרנו את בן2).

במקרה שבו P1 לא מכיל תוכן – פשוט נצא, שכן הP0 ריק וגם לא מכיל מצביעים.

לפיכך, התשובה היא עדיין N אם כי הפעם רק הריגת בן 2 תוביל להריגת כל התהליכים.

Question 2 – Interrupts, signals

1. סערת פסיקות (storm interrupt) הוא כינוי לתופעה המתרחשת כאשר קצב הגעת הפסיקות גבוה מאוד מה הבעיה במצב זה וכיצד ניתן להתמודד איתה?
הבעיה היא הoverhead הגדול שקורה בגלל הזמן שלוקח לקבל את כל הפסיקות לעומת הטיפול עצמו בפסיקות. דרך התמודדות היא בעזרת polling- הפסקת קבלת פסיקות רציפה, ובמקום זה המתנה לאירוע פסיקה בפרק זמן כלשהו.

2. נתון כרטיס רשת העובד בקצב של 20 Gbit/sec ושולח ומקבל חבילות (packets) בגודל קבוע של 4096 bit. נניח כי מערכת ההפעלה רוצה לעבוד עם הכרטיס בשיטת polling.

   a. מדוע לא כדאי לקבוע את תדר ביצוע ה- polling ל- 0.5 GHz?
   מהנתון, כרטיס הרשת עובד בקצב של כמעט Mb5 חבילות לשניה. התדר המוצע עובד על קצב של Mb512 , כלומר מהיר פי 100 ממה שהמעבד מסוגל לקלוט- לכן יתבזבזו המון מחזורי שעון על בדיקות מיותרות.

   b. מדוע לא כדאי לקבוע את תדר ביצוע ה- polling ל- 5 KHz?
   תדר תישאול זה עובד בקצב של 5Kb, תדר נמוך משמעותית מהקצב של המעבד (כלומר המעבד מסוגל לשלוח הרבה יותר חבילות בשניה) ומידע שהמעבד שולח עלול לא להיקלט בבדיקות וללכת לאיבוד.

3. בתרגול ראינו כי בפונקציה schedule לפני שניגשים לבצע פעולות על ה-runqueue משתמשים בפקודה irq_lock_spin אשר תופסת את ה-spinlock של ה-runqueue וחוסמת את הפסיקות במעבד הנוכחי.

   a. מדוע יש צורך לחסום פסיקות ולא ניתן להסתפק בתפיסת המנעול?
   כיוון שהפסיקות עצמן יכולות לגשת למנעולים ולמבני הנתונים בrunqueue, יכולות להיווצר בעיות סנכרון

   b. באיזה בעיית סנכרון היינו נתקלים אם לא היינו חוסמים את הפסיקות? יש לתאר מצב בו מתרחשת בעיה זו.
   בעיית סנכרון אפשרית היא deadlock- יכול להיות מצב בו תהליך רץ בפונ' schedule ונעל את המנעול, ולפני ששיחרר אותו הגיעה פסיקה (למשל schedule_tick) אשר גם היא מנסה לרכוש את המנעול. כעת הפסיקה יוצאת להמתנה ומחכה שייפתח המנעול, אך התהליך מחכה לפסיקה שתסיים ולכן המנעול לא ייפתח – וכך נוצר deadlock.

4. נתונה מערכת בעלת שני מעבדים B,A אשר מריצים שני תהליכים PB,PA בהתאמה. ברגע מסוים PA שולח סיגנל 1USR אל תהליך PB. הניחו כי מערכת ההפעלה אינה מבצעת החלפת הקשר ושהפסיקות והסיגנלים לא חסומים.

   a. מה פרק הזמן המקסימלי בין הרגע בו נשלח הסיגנל לרגע בו PB יתחיל לטפל בו?
   הזמן המקסימלי הוא פסיקת שעון אחת- כל פסיקת שעון שולחת את התהליך לגרעין לטיפול בפסיקה, ובכל פעם שתהליך יוצא מהגרעין הוא בודק אם ממתינים לו סיגנלים ומתפנה לטיפול בהם.

   b . תן דוגמה למצב בו PB יתחיל לטפל בסיגנל לאחר פרק זמן קצר יותר.
   יכול להיות מצב שבו התהליך כבר נמצא בגרעין כשהוא מקבל את הסיגנל, ואז כשיצא מהגרעין יתפנה לטיפול בסיגנל כפי שהוסבר קודם.

5. תארו את ההבדל בהתנהגות של הפקודה iret בעת חזרה מפסיקה מקוננת לעומת חזרה מפסיקה לא מקוננת.
   בחזרה מפסיקה לא מקוננת, ייתכן שנחזור למצב משתמש והפקודה iret תהיה אחראית על החלפת מחסניות(בעזרת שיחזור הרגיסטרים ss,esp). בחזרה מפסיקה מקוננת, החזרה תמיד תהיה לגרעין לכן לא תתבצע החלפת מחסניות.
   הפקודה תזהה באיזה מצב מדובר על ידי גישה לרגיסטר cs וקריאת משתנה ה privileged level (CPL).