TECHNION Israel Institute of Technology

Computer Science Department

Operating Systems – 234123

Homework Exercise 1 – Dry

Winter 2023

Student	id	email
Amal hihi	213519333	Hihi.amal@campus.technion.ac.il
Sarah hamou	329618169	sarah@campus.technion.ac.il

Teaching assistant in charge:

Ori Ben-Zur

Assignment Subjects & Relevant Course material

Processes and inter-process communications

Recitations 1-3 & Lectures 1-3

Submission Format

1. Only **typed** submissions in **PDF** format will be accepted. Scanned handwritten submissions will not be graded.
2. The dry part submission must contain a single PDF file named with your student IDs – **DHW1_123456789_300200100.pdf**
3. The submission should contain the following:
   a. The first page should contain the details about the submitters - Name, ID number, and email address.
   b. Your answers to the dry part questions.
4. Submission is done electronically via the course website, in the **HW1 – Dry** submission box.

Grading

1. **All** question answers must be supplied with a **full explanation**. Most of the weight of your grade sits on your **explanation** and **evident effort**, and not on the absolute correctness of your answer.
2. Remember – your goal is to communicate. Full credit will be given only to correct solutions which are **clearly** described. Convoluted and obtuse descriptions will receive low marks.

Questions & Answers

- The Q&A for the exercise will take place at a public forum Piazza **only**. Please **DO NOT** send questions to the private email addresses of the TAs.
- Critical updates about the HW will be published in **pinned** notes in the piazza forum. These notes are mandatory, and it is your responsibility to be updated.

A number of guidelines to use the forum:

- Read previous Q&A carefully before asking the question; repeated questions will probably go without answers.
- Be polite, remember that course staff does this as a service for the students.
- You're not allowed to post any kind of solution and/or source code in the forum as a hint for other students; In case you feel that you have to discuss such a matter, please come to the reception hour.
- When posting questions regarding **hw1**, put them in the **hw1** folder .

Late Days

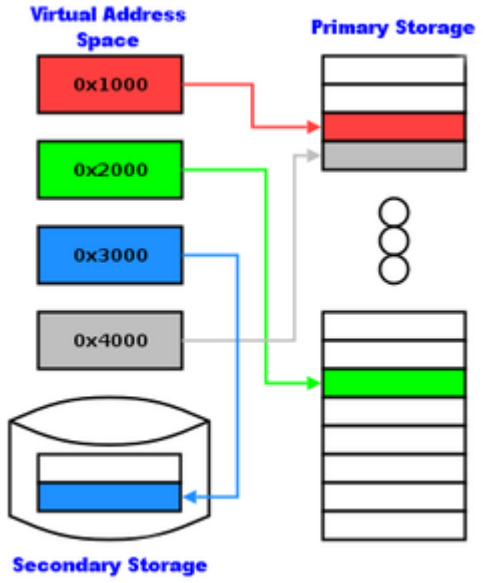
- Please **DO NOT** send postponement requests to the TA responsible for this assignment. Only the **TA in charge** can authorize postponements. In case you need a postponement, please fill out the attached form:
  https://forms.office.com/r/X8SqhiTNwe

Question 1 – Abstraction & OSs (15 points)

1. הסבירו: מהי אבסטרקציה במערכות מחשבים?
היא תהליך שבאמצעותו נתונים והוראות מוגדרים על ידי הצגה דומה למשמעות הסמנטית שלהם, תוך הסתרת פרטי המימוש שלהם.

2. מדוע אנו משתמשים באבסטרקציות במערכות הפעלה?
כדי להקל את הפיתוח של האפליקציות , מספק נוחות וניידות על ידי להציע ממשקים משמעותיים יותר, ברמה גבוהה יותר, ועל ידי הסתרת פרטי HW, מה שהופך את האינטראקציה עם HW לקלה יותר.

3. תנו דוגמה לאבסטרקציה מרכזית שמשתמשים בה במערכות הפעלה והסבירו מדוע משתמשים בה.
virtual memory זיכרון וירטואלי
הוא טכניקה לניהול והקצאה של זיכרון המחשב, המסתירה את הזיכרון הפיזי של המחשב ומדמה זיכרון רציף וגדול, ומפרידה בין ניהול הזיכרון של תהליכים שונים. כל אחת מהתוכניות המתבצעות פועלת כאילו עומד לרשותה מרחב זיכרון בגודל שהיא זקוקה לו, רציף, וללא הפרעות מתהליכים (לא מוזמנים) אחרים.
דפי הזיכרון הווירטואלי נמצאים חלקם בזיכרון ראשוני וחלקם בזיכרון משני
הרעיון של זיכרון וירטואלי הוא להוסיף עוד רמה של הפשטה בארגון הזיכרון. תהליך המנסה לגשת לזיכרון בכתובת) Xכתובת וירטואלית), ייגש בפועל לכתובת מתאימה) Yכתובת פיזית) בזיכרון הראשי של המחשב, או לכתובת Z בזיכרון משני כלשהו, בדרך כלל דיסק קשיח, שהוא זול יותר, ואז המידע המבוקש יישלף משם ויועתק לזיכרון הראשי. תרגום הכתובות מתבצע במקרים רבים בחומרה.



Question 2 – Inter-Process Communication (45 points)

נתונה התוכנית הבאה:

```
1 int main() {
2 char s[] = "A";
3 int fd = open("file", O_RDWR|O_CREAT);
4 write(fd, "B", 1);
5 close(fd);
6 fd = open("file", O_RDWR|O_CREAT);
7 close(STDIN_FILENO);
8 close(STDOUT_FILENO);
9 dup(fd);
10 dup(fd);
11 if (fork() == 0) {
12 scanf("%s", s);
13 printf("C");
14 write(STDERR_FILENO, s, 1);
15 return 0;
16 }
17 s[0] = 'D';
18 wait(NULL);
19 printf("E");
20 close(fd);
21 return 0;
22 }
```

הניחי שהקובץ file לא קיים לפני ריצת התכנית וכל קריאות המערכת מסתיימות בהצלחה .עבור כל סעיף מבין הסעיפים הבאים, ציייני (i) מה יהיה פלט התכנית למסך (ii) ומה יהיה תוכן הקובץ "file". אם יש כמה אפשרויות לתוכן הקובץ או לפלט התכנית כתבי את כולן.

1. (3 נק') עבור התכנית כמו שהיא (ללא שינויים):

פלט התכנית (2 נק'): __ B___ תוכן הקובץ: (1 נק') __BCE__

נימוק: בשורה 7 ו-8 סגרנו ערוצי in & out הסטנדרטים בשורה 9 ו-10 החלפנו אותם בקובץ fd, בשורה 18 יש wait כלומר תהליך האב ממתין לסיום תהליך הבן ,בשורה 12 אנו קוראים תו אחד מהקובץ file המכיל"B" אשר נכתב לקובץ בשורה 4, שורה 13 כותבים לקובץ את " C" , פקודת הקריאה שורה 12 קידמה את ה-headpointer ולכן הכתיבה לא תדרוס את התו " B", שורה 14 כותבים את התו שמוכל ב-("B") s לערוץ השגיאות אשר כותב למסך, בשורה 19 נכתוב "E" ל fd וסיימנו.

2. (4 נק') במידה ונסיר את שורות 5+6.

פלט התכנית (2 נק'): ___ A_____ תוכן הקובץ (2 נק'): ___BCE_____

נימוק: שורות 5 ו-6 לא מאפסות את **headerpointer** ל- 0 לאחר כתיבה של B לקובץ, כלומר הוא יצביע למיקום אחריו, ולכן scanf לא תקרא אף דבר מהקובץ , לכן scanf ייכשל ולא ידרוס את הערך שלו, ולכן ב STDERRייכתב הערך המקורי ". A" בנוסף, כמו קודם, מפני שהמחוון מצביע אחרי B לא יידרס התוכן אלא נוסיף אחריו את C, נקדם את המחוון המשותף ובסוף נכתוב את E כמו מקודם.

הכתיבה לקובץ לא תשתנה ,אותו הפלט "BCE" זאת כי הpointer ימשיך מאפה שהוא עזב בשורה 4ולכן אותה כתיבה מקודם לקובץ.

3. (4 נק') במידה ונסיר את שורה 12. (הניחי שלא בוצע שום שינוי אחר ביחס לקוד המקורי.)

פלט התכנית (2 נק'): __ _A___ תוכן הקובץ (2 נק'): ___ CE___

נימוק :__שורה 12 משנה את התו המוכל במקום הראשון ב S וגם קידום המצביע של הקובץ , ולכן מחיקת השורה גורר ש S לא יעודכן ויישאר A , השורה 13 תדרוס התו B בקובץ לתו C , ושורה 19 נכתוב E לקובץ.

4. (4 נק') במידה ונסיר את שורה 15. (הניחי שלא בוצע שום שינוי אחר ביחס לקוד המקורי.)

פלט התכנית (2 נק'): ___B____ תוכן הקובץ (2 נק'): ____BCEE ____

נימוק: כמו סעיף 1 אבל הפעם תהליך הבן ימשיך לקוד של תהליך האב וייכתב בשורה 19 לקובץ עוד E .

וב wait לא יקרה שום דבר כי אין בנים.

5. (5 נק') במידה ונסיר את שורה 18. (הניחי שלא בוצע שום שינוי אחר ביחס לקוד המקורי.)

פלט התכנית (2 נק'): ___ A\B____ תוכן הקובץ (3 נק'): __ _BCE\EC\BEC ____

נימוק: כרגע תהליך האב יוכל להיות שירוץ אחרי שורה 17 לפני תהליך הבן , ויתכן שיכתוב ל-file את התו E בדריסת B לפני שתהליך הבן יכתוב את התו C ולכן ישנם 3 אפשרויות לתוכן של הקובץ ,אם נעשה scanf ואחר כך נחזור לאב יודפס E ואז נדפיס C, או שתהליך האב יבוצע ונדרוס את B ב E ואז נדפיס C או יבוצע הבן ואז נדפיס E ונקבל BCE, אם תהליך האב דרס את התו "B" אז הפעולה בשורה 12 נכשלת ויודפס התו "A" למסך אחרת "B".

*תוכן הקבצים לא מחכים לסיום תהליך הבן, ושניהם עובדים על אותו קובץ ואותו מחוון, ולכן תוכן הקובץ לא צפוי.

6. (5 נק') במידה שנמחוק את שורות 11–16 ונוסיף ;fork() בין השורות 2 ו 3. (הניחי שלא בוצע שום שינוי אחר ביחס לקוד המקורי.)
פלט התכנית (3 נק'): ______ תוכן הקובץ (2 נק'): ___ E ___

נימוק: תהליך האב והבן פותחים את הקובץ לאחר,fork לכן כל אחד יעובד על object file שונה עם מחוון שונה. שני התהליכים רושמים" B" בתחילת הקובץ, מאפסים את המחוון בשורות 5-6 ורושמים לתחילת הקובץ את " E" שדורס את ". B" למרות שסדר הפעולות לא צפוי לנו , אבל

בסוף התהליך האחרון מי שיעבוד על הקובץ יחליף את התו בהתחלה שלו שעלול להיות או "B
"או " E" לתו" E" וייצא, לכן תוכן הקובץ יהיה .E"

כלומר לא משנה אם הם רצים במקביל , תמיד האחרון יכתוב E לתו הראשון ותמיד שניהם יכתבו לתו הראשון .

N1tero · 9m · *edited 5m*

Programming is fun, sometimes you may google stuff like:

"how to kill child and parent"
"how to kill all children in a class"
"how to re-attach a detached head"
"how to bash cat with pipe"

Question 3 – Process management (40 points)

```
int X = 1, p1 = 0, p2 = 0;
     int ProcessA() {
          printf("process A\n");
          while(X);
          printf("process A finished\n");
          exit (1);
     }
     void killAll(){
          if(p2) kill(p2, 15);
          if(p1) kill(p1, 9);
     }
     int ProcessB() {
          X = 0;
          printf("process B\n");
                 killAll();
          printf("process B finished\n");
          return 1;
     }
     int main(){
          int status;
          if((p1 = fork()) != 0)
                         if((p2 = fork()) != 0){
               wait(&status);
               printf("status: %d\n", status);
               wait(&status);
               printf("status: %d\n", status);
               } else {
                    ProcessB();
          } else {
               ProcessA();
          }
          printf("The end\n");
          return 3;
     }
```

בשאלה זו עליכן להניח כי:

1. קריאות המערכת fork() וkill() אינן נכשלות.
2. כל שורה הנכתבת לפלט אינה נקטעת ע"י שורה אחרת.
3. כאשר תהליך מקבל סיגנל x הוא מסתיים וערך היציאה שלו הוא x + 128.

עבור כל אחת משורות הפלט הבאות, סמנו כמה פעמים הן מופיעות בפלט כלשהו, נמקו את תשובתכן.

1. process A
   a. 0
   b. 0 or 1
   c. 1
   d. 1 or 2
   e. 2

נימוק: _proccessA רק מודפסת בפונקציה ()ProcessA, קוראים לפונקציה רק ע"י הבן הראשון של האב היותר ב Forkהראשון ישנם 2 מקרים :מודפס ProcessAכאשר תהליך האב יוצר את P1 ב fork , הבן P1 רץ , קורא לפונקציה ומדפיס את השורה ProcessA ואז נתקע whileבפונקציה , לאחר כך יוצר האב תהליך בן חדש P2 אשר קורא לפונקציה ProcessB השולחת סיגנל SIGKILL לתהליך הבן P1 לאחר מכן לא יתבצעו הדפסות כי לא יהיו עוד קריאות לפונקציה ProcessA

מקרה 2 : לא יודפס ProcessA האב יוצר את תהליך הבן P1 ב FORK אך הקריאה\ הדפסה מתעכבת , והאב יוצר את תהליך בן P2 אשר שולח SIGKILL לבן P1 לפני ההדפסה .

2. status: 1
   a. 0
   b. 0 or 1
   c. 1
   d. 1 or 2
   e. 2

נימוק: הפלט STATUS יכול להתקבל רק מהערך המוחזר ע"י EXIT(1) שפונקציה שתהליך הבן הראשון P1 מריץ, הוא אף פעם לא מגיע לשם כי התהליך לא משנה את הערך של X לעולם כי נכנס ללולאה אינסופית ונעצר ע"י SIGKILL, התהליך הבן השני P2 משנה את ערך זה אבל הערך לא מתעדכן אצל תהליך הבן הראשון הבן השני יחזיר ערך 3 מתוך ה MAIN ולכן בשני המקרים לא יודפס 1 ולאחר חזרת בן שני מהפונקציה ימשיך בהרצת הקוד.

3. status: 137
   a. 0
   b. 0 or 1

c. 1
d. 1 or 2
e. 2

נימוק: כאשר ProcessB ישלח סיגנל SIGKILL לתהליך הבן הראשון P1 ה status החזרה יהיה מספר הסיגנל +128 לפי ההוראות התרגיל נקודה 3 ולכן יהיה 9+128=137 וזה המקום היחיד אשר סטטוס זה יכול להתקבל

הסטטוס הזה יודפס בהכרח כי בכל מקרה נקרא לפונקציה ProcessB אשר תקרא ל KILLALL ובגלל שהבן הראשון תקוע בלולאה אינסופית תמיד האב יחכה לו עד שיסתיים ויהיה במצב ZOMBIE וזה יקרה לאחר שליחת סיגנל SIGKILL לבן P1 שאחר כך בהכרח יודפס status:137 .

4. status: 143

a. 0
b. 0 or 1
c. 1
d. 1 or 2
e. 2

נימוק:_ ערך זה יתכן להתקבל רק ע"י שליחת סיגנל 15 וזה קורה בפונקציה KILLALL אותה תהליך הבן השניP2 קוראת, אבל שורת קוד לא תתבצע כי היא נמצאת תחת if(p2) והערך p2 אצל תהליך הבן השני הוא 0 , ואת כי תהליך הבן P2 יקבל את ערך החזרה 0 מקריאת ה FORKשיצרה אותו אחרת זה היה תהליך האב .

The end

a. 0
b. 0 or 1
c. 1
d. 1 or 2
e. 2

נימוק: השורה מודפסת בשורה לפני אחרונה בפונקציה MAIN שורה תודפס לאחר שתהליך האב יבצע את הקוד ויגיע לסוף של הפונקציה , וגם בתהליך הבן השני P2 לאחר שסיימם מקריאת הפונקציה ProcessB וביצועה יחזור ל MAIN וימשיך את הקוד באותו אופן של האב , אבל תהליך בן ראשון P1 יסתיים בביצוע PROCESSA אשר נתקע שם בלולאה אינסופית עד שנהרג ע"י SIGKILL ולכן הוא לא יגיע להדפסה .