TECHNION Israel Institute of Technology

Computer Science Department

Operating Systems – 234123

Homework Exercise 1 – Dry

Spring 2023

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[1].com/r/AqYiZkU4gg

[1]

Question 1 – Abstraction & OSs (15 points)

1. הסבירו: מהי אבסטרקציה במערכות מחשבים?

אבסטרקציה במערכות מחשבים זו הפעולה של הסתרת פרטי מימוש של פונקציות או מודלים מסוימים, כך שבסופו של דבר יש לנו רק את המידע על מה הפונקציה אמורה לעשות אך לא איך היא עושה זאת.

2. מדוע אנו משתמשים באבסטרקציות במערכות הפעלה?

אנו משתמשים באבסטרקציות במערכות הפעלה ממספר סיבות, ראשית, בכך אנו יוצרים שכבת הגנה על הפונקציה, בכך שיהיה קשה לחדור אליה / להשתמש במשאבים שלה. שנית, זה מקל על פיתוח של אפליקציות בכך שהמשתמש לא צריך להיות מוסח ממימוש שלא אמור להשפיע עליו.

3. תנו דוגמה לאבסטרקציה מרכזית שמשתמשים בה במערכות הפעלה והסבירו מדוע משתמשים בה.

דוגמה אחת מרכזית לשימוש באבסטרקציה: וירטואליזציה של משאבים פיזיים)מעבד וזיכרון(עבור המשתמש, שלא אמור לדעת את פרטי המימוש)של היאך המידע של התוכנית שלו מאוחסן(ובכך הופכים את השימוש לפשוט. מלבד זאת בכך אנו מגנים על המשאב הפיזי כי אף משתמש לא יידע היכן המידע באמת קיים.

Question 2 – Inter-Process Communication (45 points)

השלימו את קוד ה-C הנתון הממשמ shell כך שיבצע את פקודת ה-bash:

/bin/prog.out < in.txt 2> err.txt > out.txt

עליכם להשתמש אך ורק בקריאות המערכת הבאות:

int **open**(const char *path, …);

int **execv**(const char *filename,

char *const argv[]);

int **close**(int fd);

- אין עליכם חובה לבדוק שקריאות המערכת צלחו.
- פתחו קבצים באמצעות הקריאה open(path,…), כלומר בפרמטר השני יש לרשום "..." (גם לקבצים קיימים וגם לחדשים).
- אין לבצע קריאות מערכת מיותרות.
- תזכורת:

"err.txt <2" מציין redirection של 2=fd (STDERR) של התהליך לכתיבה לקובץ err.txt.

```
pid_t pid = fork();

if (pid == 0) {

    close(0);
    close(1);
    close(2);
    open("in.txt", "...");
    open("out.txt", "...");
    open("err.txt", "...");
    chr* args[] = {"/bin/prog.out", NULL};
    execv(args[0], args);

} else {

    wait(NULL);

}
```

שקופיות
22,43
במצגת 3

2. (7 נקודות)

לפניכם קטע קוד המשתמש ב-pipes.

```
1  #include <stdlib.h>
2  #include <unistd.h>
3  #include <stdio.h>
4
5  int main() {
6    int fd[2];
7    int count;
8    int id;
9    pipe(fd);
10   pid_t p = fork();
11   count = get_random_between(1, 1000000);//gets random integer in [1,1000000]
12   if (p == 0) {
13     id = 234123;
14     for(int i = 0; i < count; i++)
15     {
16    write(fd[1], (void*)(&id), sizeof(int));
17     }
18   } else {
19     close(fd[1]);
20     for(int i = 0; i < count; i++)
21    {
22    if(read(fd[0], (void*)(&id), sizeof(int)) > 0)
23       {
24       printf("My favorite course, %d\n", id);
25       }
26       else
27       {
28             break;
29       }
30    }
31  }
32   return 0;
33 }
```

בשני הסעיפים הבאים, הקף את התשובה הנכונה: (x21 = 2 נקודות)

אם היינו מחליפים את שורות 10 ו-11, ערכי המשתנה count בשני התהליכים -

הקף: בהכרח זהים \ בהכרח שונים \ ייתכן ששונים וייתכן שזהים

אם **לא** נחליף את שורות 10 ו-11, ערכי המשתנה count בשני התהליכים –

הקף: בהכרח זהים \ בהכרח שונים \ ייתכן ששונים וייתכן שזהים

האם הקוד תקין? (5 נקודות)

הקיפו: כן \ לא

אם בחרתם לא, תנו דוגמא לתרחיש הבעייתי ביותר האפשרי:

חלק 2: (10 נקודות)

לפניכם קטע קוד:

```
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>


void fpe_catcher (int signum) {
  printf("Hello\n");
  exit(0);
}
int main() {
  signal(SIGFPE, fpe_catcher);
```

```
  int x = 234123 / (0);
  printf("Hi\n");
  while(1);
  return 0;
}
```

תזכורת:

SIGFPE הוא הסיגנל המתאים לשגיאות אריתמטיות, כמו חלוקה ב-0 (גם במקרה של מספרים שלמים).

1. בחרו באפשרות הנכונה בנוגע לריצת הקוד: (4 נקודות)

1. יודפס קודם "Hi" ואז "Hello".
2. יודפס רק "Hello".
3. יודפס רק "Hi".
4. לא יודפס כלום.
5. תשובות i,ii אפשריות.

נמקו:

2. כעת נניח שבזמן כלשהו של ריצת הקוד הנ"ל, תהליך אחר שולח את הסיגנל SIGFPE לתהליך המריץ את הקוד.

עבור כל אחד מהתרחישים הבאים, הכריעו האם הוא **אפשרי** או לא, ונמקו בקצרה: (4=x22 נקודות)

- יודפס "Hello" פעמיים.

הקיפו: כן \ לא

נימוק:

- לא יודפס "Hello" בכלל.

הקיפו: כן \ לא

נימוק:

3. תארו את ריצת התכנית במידה והיינו מסירים את פקודת ה-exit(0): (2 נקודות)

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:_____
_____.

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

נימוק:_____
_____.

3. status: 137
   a. 0
   b. 0 or 1
   c. 1
   d. 1 or 2
   e. 2

נימוק:_____
_____.

4. status: 143

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

נימוק:_____
_____.

5. The end

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

נימוק:_____
_____.