אלעד ישראל 313448888 קבוצת תרגול 05

הערה: ע"פ הוראות התרגיל ניתן להניח שקריאות לsystem calls ובפרט fork תמיד מצליחות.

שאלה 1

כמה תהליכים חדשים (כלומר לא כולל התהליך הראשי ממנו הפונקציה נקראה) נוצרים במהלך הריצה של התוכנית הבאה? **הסבירו תשובתכם.**

```
1. void main()
2. {
3. int a = 0;
4. a += (fork() != 0) ? 2 : 3;
5. if (a == 2) fork();
6. a++;
7. if (a == 3) fork();
8. }
```

תשובה 1:

(*הערה*- סימוני הבנים במספרים ותיאור התהליך באופן פרוצדורלי נעשה לשם הנוחות. בפועל סדר היצירה והביצוע יכול להיות שונה)

בשורה 4 מתבצע fork, נוצר בן '1'.

עבור האב: fork יהיה הPID של בן '1', כלומר שונה מ0, ולכן לa יתווסף 2.

האב יכנס לif הראשון ויעשה שוב fork. נוצר בן '2'. לa יתווסף 1 ויהיה 3.

כעת האב יכנס גם לif השני ושוב יעשה fork. נוצר בן '3' שמיד ימות.

בסה"כ יוצרו 3 תהליכים חדשים.

עבור '1': נוצר בשורה 4. fork יחזיר 0, כלומר לa יתווסף 3(המאותחל ל0 כי תהליכים לא חולקים זיכרון). ולכן לא ניכנס לשורה 5. בשורה 6 יתווסף 1 לa וערכו יהיה 4, ולכן לא ניכנס ל7 ונסיים.

עבור '2': נוצר בשורה 5. a היה 2 ובשורה 6 יהפוך ל3. כלומר ניכנס גם לשורה 7 ונעשה fork- בן '4' שמיד ימות.

לסיכום, ייוצרו 4 תהליכים חדשים, חוץ מהאב.

שאלה 2

מה תהיה התוצאה של הרצת קטע הקוד הבא? **הסבירו תשובתכם.**

הניחו שה PID של האבא הוא 168, ואילו של הבנים שלו 768,769,770 וכו'.

```
1.  #include <sys/types.h>
2.  #include <sys/wait.h>
3.  #include <unistd.h>
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.  int main()
7.  {
8.        int x = 150;
9.        printf("PARENT: x is %d\n", x);
10        printf("PARENT: forking...\n");
11        pid_t pid = fork();
12        printf("PARENT: forked...\n");
13        if (pid == 0)
14        {
15              printf("CHILD: happy birthday\n");
16              x *= 2;
17              printf("CHILD: %d\n", x);
18        }
19        else
20        {
21              wait(NULL);
22              printf("PARENT: child completed\n");
23              x *= 3;
24              printf("PARENT: %d\n", x);
25        }
26        return EXIT_SUCCESS;
27  }
```

תשובה 2:

שורה 9 תדפיס "PARENT: x is 150".

שורה 10: "PARENT: forking...".

שורה 11 תיצור בן.

שורה 12 תודפס ע"י הבן שנוצר וגם ע"י האב(בסה"כ פעמיים), ומכיוון שיש פה 2 תהליכים הכותבים בצורה לא מתואמת ועקב אופטימיזציות המעבד, ייתכן שיודפס בצורה משובשת. כלומר ייתכן שהפלט הראשון יודפס לפני השני, תוך כדי השני(בכל הצורות האפשריות), או אחרי השני.
ולכן פלט זה אצל האב יכול להיות מודפס משובש עם פלט הבן בהמשך הקוד עד שהבן ימות.

עבור האב: pid הוא 768(מוחזר מfork). כלומר ייכנס לשורה 20 ואז 21 ויחכה למות הבן.

עבור הבן: pid הוא 0(מוחזר מfork). לכן יכנס לif וידפיס "CHILD: happy birthday".
[שורה זו עלולה להיות מודפסת לפני/עם/אחרי שורה 12 אצל האב]

לאחר מכן יכפיל את x להיות 300.

ואז ידפיס "CHILD: 300", ויצא בהצלחה.
[שורה זו עלולה להיות מודפסת לפני/עם/אחרי שורה 12 אצל האב]

בחזרה לאב(חוזר לעבוד לאחר מות הבן):

ידפיס "PARENT: child completed"

x יכפיל את עצמו ב3, ויהיה 450(היה 150 אצל האב).

ידפיס: "PARENT: 450".

בסה"כ הפלט שיתקבל הוא:

```
PARENT: x is 150
PARENT: forking...
PARENT: forked...
```

[שורה זו תודפס ע"י הבן שנוצר וגם ע"י האב(בסה"כ **פעמיים**), ומכיוון שיש פה 2 תהליכים הכותבים בצורה לא מתואמת ועקב אופטימיזציות המעבד, ייתכן שיודפס בצורה משובשת. כלומר ייתכן שהפלט הראשון יודפס לפני השני, תוך כדי השני(בכל הצורות האפשריות), או אחרי השני.
ולכן פלט זה אצל האב יכול להיות מודפס משובש עם פלט הבן בהמשך הקוד עד שהבן ימות.]

```
CHILD: happy birthday
```

[שורה זו עלולה להיות מודפסת לפני/עם/אחרי שורה 12 אצל האב]

```
CHILD: 300
```

[שורה זו עלולה להיות מודפסת לפני/עם/אחרי שורה 12 אצל האב]

```
PARENT: child completed
PARENT: 450
```

שאלה 3

מה תהיה התוצאה של הרצת קטע הקוד הבא? **הסבירו תשובתכם.**

```
1.  #include <sys/types.h>
2.  #include <sys/wait.h>
3.  #include <unistd.h>
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.
7.  int main()  {
8.          pid_t child;
9.          int status;
10          child = fork();
11          switch (child) {
12              case -1:
13                  perror("fork");
14                  exit(1);
15              case 0:
16                  printf("quitting\n");
17                  _exit(2);
18              default:
19                  wait(&status);
20                  printf("%d\n",WEXITSTATUS(status));
21                  break;
22          }
23          return 0;
24  }
```

תשובה 3:

בשורה 10 נעשה `fork`.

תהליך האב: יכנס ל`default`(כי נתון שלא תקרה שגיאה), ויחכה לבן ולסטטוס הסיום שלו.

תהליך הבן: יכנס ל`case 0`: ידפיס "`quitting`"(כי למרות ש`_exit` לא בהכרח עושה `flush` ל`stdout`, ירידת שורה בהדפסה כן עושה `flush`),
ויצא עם קוד 2.

נחזור לתהליך האב: ידפיס את סטטוס הסיום של הבן: "2".
יצא מה`switch-case` ויסיים.

בסה"כ:

```
quitting
2
```

שאלה 4

מה הם כל הפלטים האפשריים של התוכנית הבאה? **הסבירו תשובתכם.**

```
1. #include <sys/types.h>
2. #include <sys/wait.h>
3. #include <unistd.h>
4. #include <stdio.h>
5. #include <stdlib.h>
6.
7. int main()
8. {
9.      int value = 3;
10      if (fork() != 0)
11      {
12           wait(&value);
13      }
14      else
15      {
16           exit(value);
17      }
18
19      value = WEXITSTATUS(value);
20      value++;
21
22      printf("%d", value);
23      return value;
24 }
```

תשובה 4:

בשורה 10 נבצע `fork`.

תהליך האב: יכנס ל`if` ויחכה לבן עד שיסיים.

תהליך הבן: יכנס ל`else` ויצא עם קוד 3.

תהליך האב: בשורה 19 יפרש את קוד היציאה של הבן ויכניס 3 ל`value`.

יוסיף 1 ל`value`. כלומר `value=4`.

ידפיס את הערך – "4", ויצא.

בסה"כ:

4