ב"ה

תרגיל מס' 2 – תהליכים

הוראות הגשה

- שאלות בנוגע לתרגיל יש לכתוב בפורום הקורס במודל.
  - בקשות פרטיות ניתן לשלוח למייל הקורס os.89231@gmail.com
- מועד אחרון להגשה 28/03/19 23:59.

- **לתרגיל זה יש שני חלקים. חלק א' מעשי וחלק ב' תאורטי.**
  - יש לשלוח את הקבצים לפני חלוף התאריך הנקוב לעיל באמצעות האתר: https://submit.cs.biu.ac.il/cgi-bin/welcome.cgi

עבור חלק א':

- יש להגיש קובץ c. יחיד. יש לרשום שם מלא ות.ז. בראש הקובץ.
- **חובה לבדוק כל פונקציה האם היא הצליחה או לא**, אם היא לא הצליחה יש לתת הודעה מתאימה ל STDERR.
- יש לוודא שהתרגיל מתקמפל ורץ על ה U2 ללא שגיאות/אזהרות.
- שימו לב להערות בסוף התרגיל
- מצ"ב קובץ jobs.pdf המכיל הסברים על פקודות בנושא בקרת תהליכים – יעזור לכם לפתרון.

עבור חלק ב':

- יש להגיש קובץ יחיד בפורמט pdf. יש לרשום שם מלא ות.ז. בראש הקובץ.
- **תשובות ללא נימוק לא יקבלו ניקוד !**

את שני הקבצים נא הגישו יחד (אפשר בזיפ).

- להזכירכם, העבודה היא אישית. "עבודה משותפת" דינה כהעתקה.

בהצלחה !

תהליכים – חלק א'

הנחיות עבור חלק א'

- שם התרגיל: ex2
- שם קובץ מקור (source file) שיש לשלוח: ex2.c
  - שימו לב שלאחר ההגשה עליכם לקבל רק מייל המאשר את ההגשה (ללא חיווי על תקינות).
- שימו לב להנחיות המופיעות בהוראות ההגשה

רקע:
בתרגיל זה תכתבו תוכנית בשפת C שתממש shell. התוכנית תציג על המסך סמן (=prompt) ותאפשר למשתמש להקליד פקודות ב unix (לדוגמא ls, cat, sleep). לאחר לחיצה על ENTER, תבוצע הפקודה שהוקלדה בתהליך נפרד ויוצג prompt חדש, עבור פקודה חדשה. אם לא כתוב אחרת, תהליך האבא יחכה (wait) לסיום תהליך הבן (exit) לפני שימשיך. אם בסוף הפקודה המשתמש הכניס את התו & ה shell יריץ את הפקודה ברקע (background).

דרישות:
1) המשתמש ב- shell יוכל להזין כל פקודה פשוטה ב-unix. אין צורך לזהות פקודות מורכבות המכילות pipe או redirection, אולם יש לאפשר הזנת ארגומנטים לפקודה.
2) לאחר רישום הפקודה ולחיצה על ENTER, יציג ה-shell למסך את ה-PID של התהליך החדש.
3) הקשת הפקודה jobs ב shell תציג את רשימת הפקודות הרצות ברקע ברגע הרצת הפקודה. כל פקודה בשורה נפרדת: קודם ה PID אחרי זה רווח ואז שם הפקודה.

הנחיות:
ה- shell שאותו תכתבו לא ינסה להבין את הפקודות שנותן המשתמש, אלא יעביר את הפקודה והארגומנטים למערכת בעזרת קריאה ל-execv (או כל מימוש של exec שנוח לכם).
כזכור, אין החזרת שליטה לתכנית לאחר קריאה ל-execv, ולכן יש ליצור תהליך חדש (fork) לפני הקריאה ל-exec. הקריאה ל-exec תבוצע בתהליך הבן.
אם המשתמש הכניס פקודה שרצה ברקע (באמצעות &), אין להמתין לסיום תהליך הבן, אלא יש להציג על המסך את ה PID של התהליך המבוצע ברקע ולאפשר הזנת פקודות נוספות באופן מיידי.
יש לשמור במערך את רשימת הפקודות המבוצעות בכדי להציגן בעת הרצת 'jobs'.
הפקודה jobs תבוצע בחזית (foreground) ולא ברקע כשאר הפקודות.

הערות:

- ניתן להניח מספר מקסימלי של ארגומנטים בפקודה (512). וכן ניתן להניח אורך מקסימלי של מערך ה jobs.
- הפקודה execv מקבלת מערך של מחרוזות. התא הראשון יכיל את הפקודה לביצוע ושאר התאים יכילו את הארגומנטים. כדי לפרק את שורת הקלט למילים ניתן להשתמש ב-strtok. לדוגמא אם המשתמש הכניס את הפקודה ls –l, המערך יהיה:

```
args[0]="ls"
args[1]="-l"
args[2]=NULL
```

- לאחר סיום פקודה, יש להוציאה ממערך הפקודות (=רשימת ה-jobs).
- שימו לב לתמיכה בפקודה cd. במימוש הרגיל פקודה זו לא תוציא את התוצאה הרצויה (למה?) ולכן עליכם לממש אותה כפקודה פנימית.

- במקרה זה יש להדפיס את ה PID של התהליך הקורא (כלומר האבא)
- שימו לב ששליחת cd לרקע (&) היא חסרת משמעות ולכן מקרה זה לא ייבדק.

- פקודות built-in אלו פקודות שממומשות באופן עצמאי ע"י ה shell. פקודת built-in נוספת שיש לממש היא exit. כלומר פקודות ה built-in היחידות שיש לממש בתרגיל זה הן cd ו exit.
  - במקרה זה יש להדפיס את ה PID של התהליך הקורא
  - אין צורך לממש את הפקודה help למרות שגם היא built-in.
  - שימו לב שהפקודה man אינה built-in ולכן יש לתמוך בה
- במצב שקריאת מערכת נכשלה יש להדפיס הודעת שגיאה בעזרת הפנייה ל file descriptor מספר 2 (stderr) באופן הבא: fprintf(stderr, "Error in system call")
  - במצב של שגיאות אחרת שנובעות מהשימוש בפקודות עצמן, אין צורך להדפיס שום דבר למסך.

דוגמת ריצה:

```
>
>
> ls –1 /home
29543
mike/
amir/
oren/
> jobs
> sleep 5000 &
29545
> sleep 100
29547
> cat hello.doc
29549
hello there.
This is the contents of the file hello.doc
> jobs
29545 sleep 5000
> sleep 100
29551
> cat hello.doc
29553
hello there.
This is the contents of the file hello.doc
>
```

תהליכים – חלק ב'

הנחיות עבור חלק ב'

- שם התרגיל: ex2
- שם קובץ שיש לשלוח: ex2.pdf
  - שימו לב שלאחר ההגשה עליכם לקבל רק מייל המאשר את ההגשה (ללא חיווי על תקינות).
- ניתן להניח בכל השאלות שכל הקריאות ל system calls מצליחות.
- שימו לב להנחיות המופיעות בהוראות ההגשה

שאלה 1

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

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

void main()
{
	int a = 0;
	a += (fork() != 0) ? 2 : 3;
	if (a == 2) fork();
	a++;
	if (a == 3) fork();
}
```

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'.

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
	int x = 150;
	printf("PARENT: x is %d\n", x);
	printf("PARENT: forking...\n");
	pid_t pid = fork();
	printf("PARENT: forked...\n");
	if (pid == 0)
	{
		printf("CHILD: happy birthday\n");
		x *= 2;
		printf("CHILD: %d\n", x);
	}
	else
	{
		wait(NULL);
		printf("PARENT: child completed\n");
		x *= 3;
		printf("PARENT: %d\n", x);
```

```
        }
        return EXIT_SUCCESS;
}
```

שאלה 3

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

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main()  {
        pid_t child;
        int status;
        child = fork();
        switch (child) {
             case -1:
                  perror("fork");
                  exit(1);
             case 0:
                  printf("quitting\n");
                  _exit(2);
             default:
                  wait(&status);
                  printf("%d\n",WEXITSTATUS(status));
                  break;
        }
        return 0;
}
```

שאלה 4

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

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main()
{
        int value = 3;
        if (fork() != 0)
        {
                wait(&value);
        }
        else
        {
                exit(value);
        }

        value = WEXITSTATUS(value);
        value++;

        printf("%d", value);
        return value;
}
```