ב"ה

תרגיל מס' 2 – תהליכים

הוראות הגשה

- שאלות בנוגע לתרגיל נא לשלוח למייל הקורס os.89231@gmail.com
- מועד אחרון להגשה 10/04/18 23:59.

- **חלק א'**
  - יש לשלוח את הקבצים לפני חלוף התאריך הנקוב לעיל באמצעות האתר: https://submit.cs.biu.ac.il/cgi-bin/welcome.cgi
  - **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.

- **חלק ב'**
  - יש לשים את התרגיל **מודפס** בתא דואר מספר 43 (של פריאל) או תא דואר מספר 29 (של מיכל) שנמצא בקומה 1- של בניין 216 (ליד מדור משכורת) לפני חלוף התאריך הנקוב לעיל.
  - תרגילים מוגשים בכתב יד לא יתקבלו.
  - **תשובות ללא נימוק לא יקבלו ניקוד !**
  - ניתן להניח בכל השאלות שפונקציית ה fork מצליחה.

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

בהצלחה !

תהליכים – חלק א'

הנחיות עבור ex2

- שם התרגיל: ex2
- שם קובץ מקור (source file) שיש לשלוח: ex2.c
- 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.

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:

- 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.
- במצב שקריאת מערכת נכשלה יש להדפיס הודעת שגיאה בעזרת הפנייה ל file descriptor מספר 2 (stderr)

דוגמת ריצה :

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

תהליכים – חלק ב'

הנחיות מופיעות בהוראות ההגשה

שאלה 1

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

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

      value = WEXITSTATUS(value);
      value++;

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

שאלה 2

מה יהיה הפלט של התוכנית הבאה? **הסבירו תשובתכם.**

```
int value = 5;
int main()
{
      pid_t pid;
      pid = fork();

      {
            if (pid == 0)
            {
                  value += 15;
                  return 0;
            }
            else
            {
                  wait(NULL);
                  printf("Parent: value = %d\n", value);
                  return 0;
            }
      }
}
```

שאלה 3

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

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

שאלה 4

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

```
void main()
{
      for (int i = 1; i <= 5; i++)
      {
            fork();
            if (i <= 2) fork();
      }
}
```