ב"ה

תרגיל מס' 1 – תהליכים

הוראות הגשה

- שאלות בנוגע לתרגיל יש לכתוב בפורום הייעודי לתרגיל במודל.
- **מועד אחרון להגשה: 6.4.21**
- יש לשלוח את הקבצים לפני חלוף התאריך הנקוב לעיל באמצעות האתר: https://submit.cs.biu.ac.il/cgi-bin/welcome.cgi

- **יש לבצע בדיקות לכישלון/הצלחה של פונקציות**, ובמקרה של כשלון להדפיס הודעה מתאימה – ראו פירוט בתרגיל.
- **יש לציין בשורה הראשונה בתרגיל שם מלא ותעודת זהות, עפ"י הפורמט הבא:**

// Israel Israeli 123456789

יש להקפיד מאוד על היצמדות לפורמט המתואר. שימו לב לרווח בין // לבין השם הפרטי.

- יש לוודא שהתרגיל מתקמפל ורץ על שרת ה-planet ללא שגיאות ו/או warnings.
- מצ"ב קובץ jobs.pdf המכיל הסברים על פקודות בנושא בקרת תהליכים – יעזור לכם להבנה טובה יותר של הפקודות הרלוונטיות. שימו לב, הקובץ מתאר את ההתנהגות הצפויה במערכות Linux. מצופה מכם לעבוד עפ"י ההוראות במסמך התרגיל, ולא עפ"י ההתנהגות שמתוארת ב-jobs.pdf.
- עבודה עם קבצים (יצירה, כתיבה, קריאה) לצרכים פנימיים של התוכנית - אסורה.
- יש להגיש קוד מתועד (באנגלית בלבד. אין לכלול תווים בעברית בקובץ הקוד).
- להזכירכם, העבודה היא אישית. "עבודה משותפת" דינה כהעתקה.

בהצלחה!

תהליכים

- שם התרגיל: ex1
- שם קובץ מקור (source file) שיש לשלוח: ex1.c
  - שימו לב שלאחר ההגשה באחריותכם לוודא שקיבלתם מייל המאשר את ההגשה.
  - תרגיל שלא התקבל מייל בהגשתו יחשב כלא הוגש.
  - **שאלות בנוגע לתרגיל יש לפרסם בפורום המתאים במודל בלבד.**

מצורפת דוגמת הרצה מפורטת בעמוד האחרון של התרגיל.

רקע

בתרגיל זה תכתבו תוכנית בשפת C שתממש shell. התוכנית תציג על המסך סמן (prompt=) ותאפשר למשתמש להקליד פקודות ב-Linux (לדוגמא ls, cat, sleep).
לאחר לחיצה על ENTER, תבוצע הפקודה שהוקלדה.
דוגמא להתנהגות התוכנית, כאשר ה-prompt מסומן ע"י התו $ (לאחריו מופיע רווח):

```
chenroz@ubuntu:~/Desktop$ gcc ex1.c
chenroz@ubuntu:~/Desktop$ ./a.out
$ ls
a.out  directoryOnDesktop  ex1.c  fileOnDesktop.txt
$ ls -l
total 32
-rwxr-xr-x 1 chenroz chenroz 13280 Mar 18 14:59 a.out
drwxrwxr-x 3 chenroz chenroz  4096 Mar 18 14:45 directoryOnDesktop
-rw-r--r-- 1 chenroz chenroz  7834 Mar 18 14:55 ex1.c
-rw-r--r-- 1 chenroz chenroz    41 Mar 18 14:57 fileOnDesktop.txt
```

התוכנית שלכם כאמור תריץ את הפקודה שהוקלדה.
בחלק מהמקרים היא תעשה זאת ע"י יצירת תהליך בן שיבצע את הפקודה.
בשאר המקרים (עבור פקודות המכונות built-in commands) – היא (תוכנית ה- shell שתכתבו) תממש את הפקודה בעצמה, ללא יצירה של תהליך בן.

פקודה אשר איננה פקודת built-in, כלומר כזו שמיועדת להיות מורצת ע"י יצירת תהליך בן, יכולה להיות מורצת באחת משתי הדרכים הבאות:

1. Foreground – במקרה זה, תהליך ה-shell שתכתבו (שמהווה תהליך אב) יקרא את הפקודה שהוקלדה ע"י המשתמש ויצור תהליך בן שיבצע אותה. תהליך האב

יחכה לסיום תהליך הבן לפני שימשיך לקרוא פקודות נוספות מהמשתמש (כלומר, לא יוצג prompt חדש מיד אלא רק לאחר סיום ריצת תהליך הבן).

2. Background – גם במקרה זה ייווצר תהליך בן, אך כאן תהליך הבן יורץ ברקע, מבלי שתהליך האב יחכה לו. prompt חדש יוצג, לצורך קליטת פקודה חדשה מהמשתמש, ללא תלות בשאלה האם תהליך הבן כבר הסתיים או לא.

כלומר, אם המשתמש הכניס פקודה להרצה ב-background יש לאפשר המשך הזנת פקודות באופן מיידי (ולא רק אחרי שתהליך הבן יסיים).
כדי לסמן הרצה ב-background המשתמש יקליד בסוף הפקודה את התו & (אשר יופרד באמצעות רווח משאר הפקודה).

לדוגמא, עבור הרצת הפקודה ls ב-foreground נכתוב

```
ls
```

ועבור הרצתה ב-background נכתוב

```
ls &
```

שימו לב, הפקודות צריכות להיקלט במסגרת התוכנית עצמה ולא כארגומנטים לתוכנית.
המשתמש ב- shell יוכל להזין כל פקודה פשוטה ב-Linux.
אין צורך לזהות פקודות מורכבות המכילות pipe או input/output redirection (למשל הפניית פלט לקובץ בעזרת >), אולם יש לאפשר הזנת ארגומנטים לפקודה עצמה. כמו כן, אין צורך לתמוך בפקודות השמה ב-shell.

עבור פקודות שאינן built-in, ה- shell שאותו תכתבו לא ינסה להבין את הפקודות שמזין המשתמש, אלא יעביר את הפקודה והארגומנטים למערכת בעזרת קריאה לפונקציה ממשפחת exec. לכן לא מצוינת בתרגיל רשימת פקודות בהן צריך לתמוך (זו לא טעות).
ניתן להניח שהנתיב בו מופיע קובץ ההרצה עבור הפקודה הרלוונטית יופיע בתוך משתנה הסביבה PATH - חשבו אילו פונקציות ממשפחת exec מתאימות תחת הנחה זו.

כזכור, אין החזרת שליטה לתוכנית לאחר קריאה (שהצליחה) ל-exec, ולכן יש ליצור תהליך חדש (fork) לפני הקריאה ל-exec, כך שהקריאה ל-exec תבוצע מתהליך הבן ולא מתהליך האב.
בפקודות built-in כן יש צורך במימוש שלכם של הפקודה, ושם תינתן רשימה של פקודות דרושות, אבל כאמור, את שאר הפקודות אתם לא צריכים לממש אחת-אחת, אלא להשתמש בפונקציה ממשפחת exec.

ועכשיו לפקודות ה-built-in -

כאמור, פקודות built-in אלו פקודות שממומשות באופן עצמאי ע"י תוכנית ה-shell **ואינן יוצרות תהליך בן**. בתרגיל עליכם לממש ארבע פקודות built-in: jobs, history, cd, exit.

להלן פירוט:

1. jobs

תציג את רשימת הפקודות הרצות ברגע ברקע (background), כפי שהוזנו ע"י המשתמש ברגע הרצת הפקודה (ללא ה-&), לפי סדר כרונולוגי של הכנסתן ע"י המשתמש (מהמוקדם למאוחר).

2. history

תציג את רשימת כל הפקודות שהמשתמש הכניס במהלך ריצת התוכנית, בסדר כרונולוגי מהמוקדם למאוחר, כולל פקודות שנכשלו.
לאחר כל פקודה יש להדפיס RUNNING או DONE, כאינדיקציה האם ריצתה הסתיימה או לא. חשוב לשים לב ש-RUNNING או DONE מתייחסים לשאלה האם הפקודה עצמה סיימה, ולא האם התהליך שהריץ אותה סיים.
ראו פורמט הדפסה בדוגמא שבסוף התרגיל (גם כאן בלי ה-& עבור הרצה ברקע במידה והוזן – כמו ב-jobs).
יש להדפיס את כל הפקודות שהמשתמש הכניס בעבר, כולל jobs ו-history.
בהרצת הפקודה history, הפקודה האחרונה שתודפס תהיה בהכרח history (כלומר התייחסות הפקודה לעצמה) והיא תיחשב כ-RUNNING.

3. cd

משנה את ה-working directory (current) של התהליך. זה אותו working directory שראינו בתרגול שהפקודה pwd מדפיסה.
מעבר לכך, אין צורך לשנות את ה-prompt בהתאם או להדפיס את ה- working directory.
cd צריך לתמוך (חוץ ממעבר לתיקייה) ב-3 דגלים הבאים:
~, .. , -

בהקשר לדגלים שצוינו, יש לתמוך גם בצירופים, למשל אם נסתכל כדוגמא על .., לא מספיק לתמוך רק ב .. cd , אלא יש לתמוך גם בצירופים דוגמת:
cd ../../HI

על מנת לממש את cd עליכם להשתמש בפונקציה chdir – אשר משנה את ה-working directory של התהליך הקורא.
פונקציה שימושית נוספת היא הפונקציה getcwd - דרכה ניתן לקבל את ה-working directory הנוכחי של התהליך הקורא.

ניתן למצוא מידע עליהן בין היתר ע"י שימוש ב-man (הרצת הפקודה man או חיפוש בגוגל – בדר"כ ע"י חיפוש man COMMAND linux). שימו לב להתנהגות הפונקציות ולערכים המוחזרים מהן. ההתנסות בלמידה מ-man היא אחת ממטרות התרגיל.
בנוסף, את תיקיית הבית של המשתמש ניתן לקבל ע"י getenv("HOME")

הסיבה שהפקודה cd צריכה להיות ממומשת ע"י תוכנית ה-shell עצמה ולא ע"י תהליך בן שלה, היא שאנחנו רוצים שה-working directory ישתנה עבור תהליך ה-shell עצמו.
נשים לב שלא מצופה ששימוש ב-cd ישפיע על ה-shell המקורי במערכת לאחר סיום ריצת התוכנית, אלא ישפיע רק על ה-shell שאנחנו כותבים. אם תריצו את pwd (שמדפיסה את ה-working directory של התהליך) ב-shell המקורי שלכם לפני ואחרי סיום ריצת התוכנית – אתם צריכים לקבל תוצאה זהה, גם אם הורץ cd מתוך התוכנית שלכם. ניתן לראות זאת גם בדוגמת ההרצה שבסוף התרגיל – המיקום שלנו לפני ואחרי הרצת התוכנית לא השתנה (=תיקיית ה-Desktop).

4. <u>exit</u>
תביא ליציאה מהתוכנית, עם ערך חזרה 0.

אין צורך לממש פקודות built-in אחרות ב-Linux שאינן מופיעות ברשימה זו (כמו הפקודה help למשל).

<u>ניתן להניח שפקודות built-in לא יורצו ברקע בתהליך הבדיקה, כלומר לא יצורף אליהן & והן יורצו רק בחזית.</u>

<u>הדפסת הודעות שגיאה</u>

- עבור כשלון של exec יש להדפיס:

exec failed

- עבור כשלון של fork יש להדפיס:

fork failed

- עבור כשלון של chdir יש להדפיס:

chdir failed

- במידה והמשתמש הקליד את הפקודה cd עם יותר מארגומנט אחד, יש להדפיס:

Too many argument

- עבור כל כשלון אחר יש להדפיס:

An error occurred

מעבר לכך, אין להדפיס שגיאות נוספות מהתוכנית שלכם. מצב בו הפקודות עצמן ידפיסו שגיאה (כלומר לא עפ"י הגדרה שלכם) הוא תקין. לדוגמא:

```
chenroz@ubuntu:~/Desktop$ ls -j
ls: invalid option -- 'j'
Try 'ls --help' for more information.
```

שימו לב שגם לאחר שגיאה בהרצת פקודה, ה-shell עצמו ממשיך לקלוט פקודות נוספות.

נקודה קריטית – לאורך כל התרגיל, יש להדפיס למסך עם הפונקציה printf.
מיד לאחר כל שימוש ב-printf שלא מסתיים בתו ירידה שורה (\n), למשל בהדפסת ה-prompt, יש להוסיף את השורה הבאה:

```
fflush(stdout);
```

יש צורך ב-

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

אי הוספת השורה עלולה להביא לכתיבה מאוחרת יותר של ההדפסה שלכם למסך במידה ולא התבצע flush to buffer, וכתוצאה מכך לכישלון טסטים בבדיקה.
חשוב - אין להדפיס הדפסות נוספות מעבר למה שנדרש בתרגיל.

- ב-prompt יש להדפיס " $" (כלומר $ ואז רווח)

- ניתן להניח אורך פקודה מקסימלי של 100 תווים.
- ניתן להניח שהתוכנית לא תורץ מנתיבים שאורכם עולה על 100 תווים.
- ניתן להניח שיוכנסו בבדיקה לכל היותר 100 פקודות.
- הפקודה exec עם האות v 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
```

- ניתן להניח שלא יוכנסו רווחים בשמות תיקיות וקבצים.
- הפקודה ls ~ לא תיבדק.
- עבור פקודת echo – יכולה ב-shell לקבל מחרוזת עם גרשיים ובלי גרשיים ועדיין תדפיס את המחרוזת ללא הגרשיים. יש לטפל בשני המקרים. ניתן להניח שאם תועבר מחרוזת היא תועבר או ללא גרשיים או איתם (לא עם ', אלא רק ").

שימוש ב-fork על שרת ה-planet

גם במחשב שלכם וגם על שרת ה-planet עליכם לגלות ערנות בשימוש בפונקציה fork כדי לא ליצור מספר רב של תהליכים בצורה לא מבוקרת. בפרט, בשרת ה-planet קיימת מגבלה על מספר התהליכים לכל משתמש, כך שאם מבוצע fork אינסופי (היזהרו בשימוש בלולאות) לא תהיה אפשרות למשתמש להתחבר לשרת מחדש היות ולא יתאפשר ליצור את התהליכים הדרושים.
הפקודות השימושיות בהקשר הזה הן ps, kill, pkill
יש לקרוא את ההסבר כאן: https://support.esc.biu.ac.il/359/

השתמשו בפקודה ps (שמציגה מידע על התהליכים הרצים במערכת – ניתן לראות דוגמאות גם ב-jobs.pdf) לאחר הרצת התרגיל שלכם ולפני שאתם מתנתקים מהשרת, וודאו שהתוצאה נקייה ושאין תהליכים מיותרים.

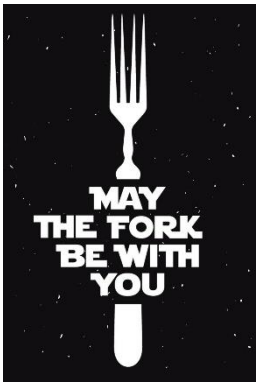
<u>לנוחיותכם, להלן רשימה חלקית ומקוצרת של נקודות חשובות שצוינו בתרגיל:</u>

- לא לשכוח לציין שם ות.ז בראש התרגיל, עפ"י הפורמט הדרוש (ראו עמוד ראשון בתרגיל).
- יש לוודא קבלת מייל עם פידבק ראשוני לאחר הגשת התרגיל.
- נקודה קריטית – printf ו-fflush – עקבו אחר ההוראות בתרגיל!
- לבדוק את התרגיל על שרת ה-planet. יש לוודא שמתקמפל ושנותן את הפלט המצופה בבדיקות שלכם. להקפיד על יצירה מבוקרת של תהליכים (שימוש ב-fork) ולקרוא את ההסבר כאן https://support.esc.biu.ac.il/359/.
- לוודא שאין שגיאות או warnings.
- אין להדפיס הדפסות נוספות מעבר למה שנדרש בתרגיל.

אין להסתפק ברשימה זו ויש לוודא מעבר על כל הוראות התרגיל, ובפרט על העמוד

דוגמת ריצה מצורפת בעמוד הבא.

בהצלחה!



דוגמת ריצה:

```
chenroz@ubuntu:~/Desktop$ gcc ex1.c
chenroz@ubuntu:~/Desktop$ ./a.out
$ ls
a.out  directoryOnDesktop  ex1.c  fileOnDesktop.txt
$ ls -l
total 32
-rwxr-xr-x 1 chenroz chenroz 13280 Mar 18 14:59 a.out
drwxrwxr-x 3 chenroz chenroz  4096 Mar 18 14:45 directoryOnDesktop
-rw-r--r-- 1 chenroz chenroz  7834 Mar 18 14:55 ex1.c
-rw-r--r-- 1 chenroz chenroz    41 Mar 18 14:57 fileOnDesktop.txt
$ jobs
$ history
ls DONE
ls -l DONE
jobs DONE
history RUNNING
$ sleep 25 &
$ sleep 1
$ cat fileOnDesktop.txt
This is the content of fileOnDesktop.txt
$ jobs
sleep 25
$ history
ls DONE
ls -l DONE
jobs DONE
history DONE
sleep 25 RUNNING
sleep 1 DONE
cat fileOnDesktop.txt DONE
jobs DONE
history RUNNING
$ jobs
$ history
ls DONE
ls -l DONE
jobs DONE
history DONE
sleep 25 DONE
sleep 1 DONE
cat fileOnDesktop.txt DONE
jobs DONE
history DONE
jobs DONE
history RUNNING
$ cd directoryOnDesktop
$ ls
innerDir
$ exit
chenroz@ubuntu:~/Desktop$
```