תרגיל מס' 1 – Bash, process, exec

- יש לרשום שם מלא ות.ז. באופן הבא:
  - בקובץ ה c יש לרשום בשורה הראשונה של הקובץ שם ות.ז בפורמט הבא:
    //Israel Israeli 123456789
  - בכל הסקריפטים ב bash shell שימו לב שהשורה הראשונה בסקריפט אמורה להתאים לכך. עבור bash, נכתוב: #!/bin/bash
    שם ות.ז יופיעו בהערה בשורה שלאחר מכן, לפי הפורמט הבא:
    #Israel Israeli 123456789
- שם ההגשה של התרגיל: tennis.sh, gccfind.sh, myshell.c
- להזכירכם, העבודה היא אישית. "עבודה משותפת" דינה כהעתקה.
- אין להדפיס שום דבר מעבר למה שנתבקש בתרגיל.
- יש לוודא שהתרגיל מתקמפל ורץ על שרת ה-planet ללא שגיאות/אזהרות.

חלק 1 – Bash חימום

סעיף א

כתבו סקריפט בשם gccfind.sh המקבל את הארגומנטים הבאים:

- נתיב לתיקיה (או שם של תיקיה במיקום הנוכחי, ניתן להניח שקיימת ושאינה מכילה רווחים בשמה)
- מילה כלשהי (ניתן להניח שאינה ריקה)

בריצת הקובץ הוא עובר על התיקיה שהתקבלה וראשית מוחק ממנה את כל הקבצים המקומפלים (ניתן להניח שכולם מסתיימים ב out.). לאחר מכן, קורא את הקבצי c בתיקייה, ומקמפל רק את קבצי ה c באותה התיקייה (שם קבצי הפלט כשם קבצי ה c שקומפלו מהם ובמקום c. הסיומת out. ראה דוג') אשר מכילים בתוכם את המילה הנתונה (לא משנה אם היא חלקה באותיות גדולות או קטנות).

דגל אפשרי (ניתן להניח שאם מופע יופיע תמיד לאחר 2 הארגומנטים הראשונים):

- דגל r- יכול להתווסף/לרדת, אם מופיע יש לבצע את אותה הפקודה שמתוארת רק באופן רקורסיבי על התיקיה הניתנת (עובר על התיקיה שהתקבלה, ולאחר מכן על כל תתי-התיקיות שלה (עד הרמה האחרונה בעץ)).

דגל אפשרי נוסף:

<u>הערות</u>:

- בקימפול יש להשתמש בדגל w- על gcc על מנת שלא יודפסו warnings.
- אפשר להניח שכל קבצי הc בתיקייה שתינתן יכולים להתקמפל ללא בעיות.
- שימו לב, מילה שמכילה את המילה שנחפש (אך ארוכה ממנה) לא תחשב.
- ניתן להניח ששמות תיקיות וקבצים לא יכילו אותיות גדולות.
- במידה ולא התקבלו מספיק ארגומנטים יודפס "Not enough parameters"
- אם תינתן המילה world, אז world. ו-world! יהיו התאמה (זה לא הכלה, אלא מופע של המילה). מקרים כמו world's לא יינתנו ולכן אין צורך לתמוך בכך.
- <u>רמז</u>: כדאי להתחיל לממש באופן שהוא לא רקורסיבי ורק לאחר מכן לממש עם הרקורסיה כאשר הקובץ יריץ גם פקודות gccfind.sh, שימו לב על מנת שנוכל להריץ את הפקודה גם אם קראו לה מחוץ לתיקייה לדוג' ע"י

  ./my_command/gccfind.sh 0 kslsssss -r

  ניתן להיעזר בארגומנט 0 שמכיל את האופן שבו קראו לנו ולקרוא שוב לgccfind.sh בתוך עצמה ע"י:

  ```
  $($0 $dirPath $wordToFind)
  ```

  ובאופן רקורסיבי לקרוא לעצמה ע"י:

  ```
  $($0 $dirName $wordToFind "-r")
  ```

דוג' (זו דוג' אחת מתחילה משמאל וממשיכה בימין):

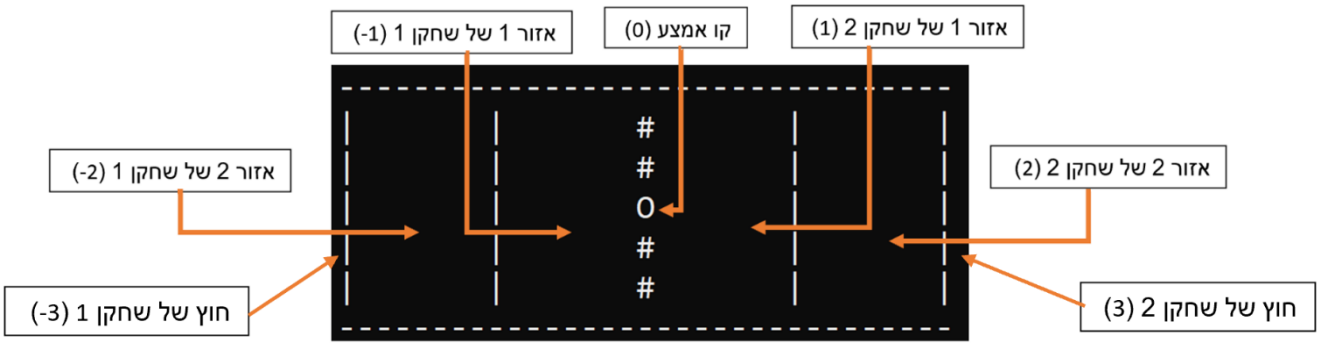
```
root@DESKTOP-HKCE1KB:~/hw1_os# tree ./0
./0
├── 1
│   ├── 3
│   │   ├── atexit_example_1.c
│   │   └── atexit_example_2.c
│   ├── 4
│   │   ├── execvp_wait_example.c
│   │   ├── fork_example_1.c
│   │   └── fork_example_2.c
│   ├── execl_example.c
│   └── execv_example.c
├── 2
│   ├── 5
│   │   ├── 6
│   │   │   ├── t2_2.c
│   │   │   └── t2_3.c
│   │   └── t2_1.c
│   └── fork_example_3.c
├── wait_example_1.c
├── wait_example_2.c
└── waitpid_example.c

6 directories, 14 files
root@DESKTOP-HKCE1KB:~/hw1_os# ./gccfind.sh ./0/1 main
root@DESKTOP-HKCE1KB:~/hw1_os# tree ./0
./0
├── 1
│   ├── 3
│   │   ├── atexit_example_1.c
│   │   └── atexit_example_2.c
│   ├── 4
│   │   ├── execvp_wait_example.c
│   │   ├── fork_example_1.c
│   │   └── fork_example_2.c
│   ├── execl_example.c
│   ├── execl_example.out
│   ├── execv_example.c
│   └── execv_example.out
├── 2
│   ├── 5
│   │   ├── 6
│   │   │   ├── t2_2.c
│   │   │   └── t2_3.c
│   │   └── t2_1.c
│   └── fork_example_3.c
├── wait_example_1.c
├── wait_example_2.c
└── waitpid_example.c

6 directories, 16 files
root@DESKTOP-HKCE1KB:~/hw1_os# ./gccfind.sh ./0/1 main -r
root@DESKTOP-HKCE1KB:~/hw1_os# tree ./0
./0
├── 1
│   ├── 3
│   │   ├── atexit_example_1.c
│   │   ├── atexit_example_1.out
│   │   ├── atexit_example_2.c
│   │   └── atexit_example_2.out
│   ├── 4
│   │   ├── execvp_wait_example.c
│   │   ├── execvp_wait_example.out
│   │   ├── fork_example_1.c
│   │   ├── fork_example_1.out
│   │   ├── fork_example_2.c
│   │   └── fork_example_2.out
│   ├── execl_example.c
│   ├── execl_example.out
│   ├── execv_example.c
│   └── execv_example.out
├── 2
│   ├── 5
│   │   ├── 6
│   │   │   ├── t2_2.c
│   │   │   └── t2_3.c
│   │   └── t2_1.c
│   └── fork_example_3.c
├── wait_example_1.c
├── wait_example_2.c
└── waitpid_example.c

6 directories, 21 files
root@DESKTOP-HKCE1KB:~/hw1_os# ./gccfind.sh ./0/1
Not enough parameters
```

```
root@DESKTOP-HKCE1KB:~/hw1_os# ./gccfind.sh ./0 mmamkmakslk -r
root@DESKTOP-HKCE1KB:~/hw1_os# tree ./0
./0
├── 1
│   ├── 3
│   │   ├── atexit_example_1.c
│   │   └── atexit_example_2.c
│   ├── 4
│   │   ├── execvp_wait_example.c
│   │   ├── fork_example_1.c
│   │   └── fork_example_2.c
│   ├── execl_example.c
│   └── execv_example.c
├── 2
│   ├── 5
│   │   ├── 6
│   │   │   ├── t2_2.c
│   │   │   └── t2_3.c
│   │   └── t2_1.c
│   └── fork_example_3.c
├── wait_example_1.c
├── wait_example_2.c
└── waitpid_example.c

6 directories, 14 files
root@DESKTOP-HKCE1KB:~/hw1_os# ./gccfind.sh ./0 child -r
root@DESKTOP-HKCE1KB:~/hw1_os# tree ./0
./0
├── 1
│   ├── 3
│   │   ├── atexit_example_1.c
│   │   └── atexit_example_2.c
│   ├── 4
│   │   ├── execvp_wait_example.c
│   │   ├── execvp_wait_example.out
│   │   ├── fork_example_1.c
│   │   └── fork_example_2.c
│   ├── execl_example.c
│   └── execv_example.c
├── 2
│   ├── 5
│   │   ├── 6
│   │   │   ├── t2_2.c
│   │   │   ├── t2_3.c
│   │   │   └── t2_3.out
│   │   ├── t2_1.c
│   │   └── t2_1.out
│   ├── fork_example_3.c
│   └── fork_example_3.out
├── wait_example_1.c
├── wait_example_2.c
├── wait_example_2.out
├── waitpid_example.c
└── waitpid_example.out

6 directories, 20 files
root@DESKTOP-HKCE1KB:~/hw1_os#
```

חלק 2 - מימוש המשחק (paper game) Tennis ב Bash

כללי המשחק:

- לוח המשחק (מגרש הטניס) מחולק ל 4 אזורים + 2 חוצים וקו אמצעי מפריד ומיוצג ע"י (-3,-2,-1,0,1,2,3) וכדור שסימונו O:

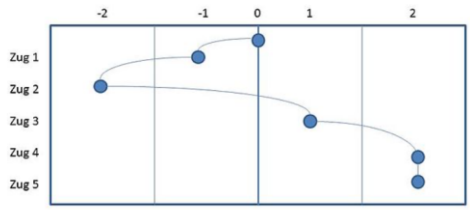


- בתחילת המשחק הכדור מתחיל על הקו האמצעי (0) וכל שחקן מתחיל עם 50 נק'. שימו לב רק בהתחלה הכדור מתחיל על הקו האמצעי וברגע שיצא ממנו לשארית המשחק הוא לא יחזור להיות על הקו האמצעי.
- כעת בכל תור שני השחקנים בוחרים מס' (בין 0 למס' הנק' הנוכחי שיש להם כולל) וחושפים אותו יחד, כעת השחקן בעל המס' הקטן ביותר "מפסיד" בסבב והכדור נע לעברו באופן הבא:
  - אם הכדור היה כבר באזור של השחקן ה"מפסיד" אזי הכדור זז אזור אחד נוסף לכיוון השחקן ה"מפסיד".
  - אחרת, הכדור באזור של השחן ה"מנצח" או בקו האמצע והכדור יעבור לאיזור ה- 1 של השחקן המפסיד.

  לאחר מכן מחסירים מהנק' של כל שחקן את המס' שהוא בחר.
- המשחק מסתיים כאשר מגיעים לאחד מהמקרים הבאים (לפי עדיפות מתחילים לבדוק מהראשון לאחרון ואם אף אחד לא מתקיים המשחק ממשיך):
  - אם הכדור באחד מה- "חוצים" (3,-3) של אחד השחקנים אזי השחקן שהכדור לא בחוץ שלו מנצח.
  - אם אחד מהשחקנים הגיע בנק' ל-0 והשחקן האחר עדיין יש נק' אזי השחקן האחר ניצח (הוספנו את החוק הנ"ל על מנת לייעל את המשחק).
  - אם שני השחקנים הגיעו ל 0 ביחד (בסוף התור) אזי השחקן שהכדור באזור שלו מפסיד (ואם הכדור עדיין על קו האמצע אזי תיקו).

דוג' למשחק מויקיפדיה (למידע נוסף: לחצו כאן):

t	Player 1 Draw $Z_{1,t}$	Player 2 Draw $Z_{2,t}$	Player 1 Status $S_{1,t}$	Player 2 Status $S_{2,t}$	Ball B_t	Comment
0			50	50	0	Start
1	5	10	45	40	-1	
2	5	10	40	30	-2	
3	15	10	25	20	1	
4	15	10	10	10	2	
5	10	10	0	0	2	Player 1 wins



התרגיל:

ממשו את המשחק Tennis (paper game) ב-Bash שם הקובץ tennis.sh.

דרישות (המחשה בדוג' הרצה):

- בכניסה למשחק ובמהלכו יופיע הלוח מאותחל (כאשר הכדור בקו האמצע) ומעליו תמיד מס' הנק' שנותרו לשחקנים.
- כעת בכל תור יש לקלוט מהמשתמשים את המס' שהם בוחרים יש לוודא שהמס' הוא בין 0 למס' הנק' שנותרו לאותו שחקן במידה והמס' חורג מהגבולות או שהקלט אינו מספר יש להדפיס את הודעת השגיאה not a valid move (הניסוח המלא ב tennis_str.txt).
- לאחר התור הראשון והלאה יופיע תמיד מתחת ללוח את המס' שבחרו השחקנים בתור האחרון שהיה.

דגשים נוספים:

- כל ה strings שתצטרכו נמצאים בקובץ tennis_str.txt תדאגו להדפיס את אותם strings בדיוק (במקומות שיש var תחליפו בשם המשתנה שתירצו להדפיס) במקומות הנכונים שימו לב strings שמכילים \n דורשים הדפסה ע"י echo -e, ובכלל כל ה strings יודפסו ע"י echo -e או echo ולא ע"י פקודה אחרת.
- את הדפסת השורה האמצעית בלוח שמכילה תמיד את הכדור יש להדפיס ע"י switch case

את אופן הרישום תוכלו ללמוד מכאן.

- את אופן קריאת המס׳ שהשחקנים בוחרים בכך תור יש לקרוא ע״י read -s כאשר הדגל s גורם לכך שהמס׳ שהשחקנים ירשמו לא יופיע בקלט שהם מכניסים על מנת שהשחקן השני לא יראה את המס׳ בזמן ההכנסה.
- על מנת לבדוק שהקלט שמכניסים לשחקנים הוא באמת מס׳ את יכולים להשתמש בהשוואת הקלט באופן הבא:

```
$var =~ ^[0-9]+$
```

זה משתמש בregex להרחבה ראו: כאן וכאן

- **המלצה:** תשתמשו בפונקציות!!!! (הרחבה על סינטקס כאן)
- מי שרוצה יכול להביט במימוש של המשחק איקס עיגול ב bash כאן, המימוש יכול לעזור באופן מימוש התרגיל ולתת רעיון טוב איך לגשת לחלק הנ״ל.

דגשים כללים לכל קבצי ה bash שתגישו:

- עליכם לכתוב את כל הסקריפטים ב bash shell. שימו לב שהשורה הראשונה בסקריפט אמורה להתאים לכך. עבור bash, נכתוב:

```
#!/bin/bash
```

- שימו לב שעל מנת להריץ את הסקריפט, עליכם לתת הרשאות מתאימות – בפרט הרשאת הרצה למשתמש שיצר את הקובץ. ניתן לעשות זאת ע״י chmod 700, או chmod u+x שרק מוסיף את הרשאת ההרצה עבור ה-owner.

דוג' הרצה (משמאל אותה הדוג' שנלקחה מויקיפדיה):

```
root@DESKTOP-HKCE1KB:~/hw1_os# ./tennis.sh
 Player 1: 50         Player 2: 50
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       O       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 45         Player 2: 40
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |   O   #       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 5
       Player 2 played: 10


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 40         Player 2: 30
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |   O   |       #       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 5
       Player 2 played: 10


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 25         Player 2: 20
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       #   O   |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 15
       Player 2 played: 10


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 10         Player 2: 10
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       #       |   O   |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 15
       Player 2 played: 10


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 0          Player 2: 0
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       #       |   O   |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 10
       Player 2 played: 10


PLAYER 1 WINS !
root@DESKTOP-HKCE1KB:~/hw1_os#
```

```
root@DESKTOP-HKCE1KB:~/hw1_os# ./tennis.sh
 Player 1: 50         Player 2: 50
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       O       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 40         Player 2: 40
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       O       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 10
       Player 2 played: 10


PLAYER 1 PICK A NUMBER:
NOT A VALID MOVE !
PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 25         Player 2: 16
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |   O   #       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 15
       Player 2 played: 24


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 13         Player 2: 2
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |   O   |       #       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 12
       Player 2 played: 14


PLAYER 1 PICK A NUMBER:
NOT A VALID MOVE !
PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 10         Player 2: 0
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       #   O   |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 3
       Player 2 played: 2


PLAYER 1 WINS !
root@DESKTOP-HKCE1KB:~/hw1_os#
```

```
root@DESKTOP-HKCE1KB:~/hw1_os# ./tennis.sh
 Player 1: 50         Player 2: 50
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       O       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 39         Player 2: 33
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |   O   #       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 11
       Player 2 played: 17


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 24         Player 2: 19
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       #   O   |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 15
       Player 2 played: 14


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 20         Player 2: 12
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |   O   #       |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 4
       Player 2 played: 7


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 13         Player 2: 10
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       #   O   |       |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 7
       Player 2 played: 2


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 8          Player 2: 6
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       #       |   O   |
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 5
       Player 2 played: 4


PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 0          Player 2: 0
 ---------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       #       |       O
 |       |       #       |       |
 |       |       #       |       |
 ---------------------------------
       Player 1 played: 8
       Player 2 played: 6


PLAYER 1 WINS !
root@DESKTOP-HKCE1KB:~/hw1_os#
```

חלק 3 – מימוש shell מינימלי

בחלק זה תכתבו תוכנית בשפת C שתממש shell. התוכנית תקרא myshell.c. התוכנית תציג על המסך סמן (=prompt) ותאפשר למשתמש להקליד פקודות ב- Linux(לדוגמא ls, cat, sleep). לאחר לחיצה על ENTER, תבוצע הפקודה שהוקלדה, בתהליך נפרד.

כמו כן הshell יוכל לקבל כל מס' של ארגומנטים כאשר כל ארגומנט (אם יש) יכיל נתיב מלא לתיקייה, כעת בריצת התוכנית כל פקודה שתיכנס לshell תוכל לרוץ אם היא פקודה שידועה כבר כמו (ls) או אם היא פקודה שנמצאת באחת מהתיקיות שנכנסו לדוג' (gccfind.sh במידה ויש תיקיי שמכילה אותה). (רמז: ...אהמ אהמ...משתני סביבה...אהמ אהמ...)

דרישות:

- המשתמש ב shell -יוכל להזין כל פקודה פשוטה ב .unix-אין צורך לזהות פקודות מורכבות המכילות pipe או input/output redirection, אולם יש לאפשר הזנת ארגומנטים לפקודה.
- כל הפקודות יורצו ב Foreground כלומר, הפקודות יורצו ע"י הבן ו exec ולאחר מכן האבא יחכה שהבן יחזור לפני שניתן יהיה לקלוט את הפקודה הבאה.
- הקלדת הפקודה history ב shell-תציג את רשימת כל הפקודות שהמשתמש הכניס במהלך ריצת התוכנית. בהרצת הפקודה history הפקודה האחרונה שתודפס תהיה בהכרח history (כלומר התייחסות הפקודה לעצמה). כמו כן, משמאל לכל פקודה שמודפסת יודפס ה PID של ה process שהריץ אותה.
- מימוש פקודות built in:

פקודות built-in אלו פקודות שממומשות באופן עצמאי ע"י ה shell-ואינן יוצרות תהליך בן. בתרגיל עליכם לממש 3 פקודות כאלה: exit, cd, history:

  - cd: עליכם לממש אותה כפקודה פנימית ע"י chdir אין צורך לתמוך ב – cd ו ~ cd במידה ו chdir-נכשלה יש להתייחס כאילו פקודת מערכת נכשלה (הסברים בנוגע להתייחסות לשגיאות מופיעים בהמשך התרגיל).

    לא מצופה ששימוש ב cd-ישפיע על ה shell-המקורי במערכת לאחר סיום ריצת התוכנית. אם תבדקו את pwd ב shell-המקורי שלכם לאחר סיום ריצת התוכנית – לא צריך להיות שינוי כתוצאה מהרצת cd מהתוכנית שלכם.
  - exit: כאשר נרשום פקודה זו ב shell התוכנית תסתיים ונצא מה shell.
- כאשר אתם משנים משתנה סביבה, יש רק להוסיף למשתנה הסביבה מידע ולא לדרוס אותו, כמו כן יש לדאוג שכל שאר המשתנה סביבה שמועברים לתוכנית בזמן הריצה (שלא שיניתם) עדיין נשארים זהים כלומר אם ארשום env ב shell אוכל לראות את כל המשתנה הסביבה המקוריים + השינוי שביצעתם למשתנה סביבה מסויים.

הנחיות והערות נוספות:

- עבור פקודות שאינן built-in, ה shell-שאותו תכתבו לא ינסה להבין את הפקודות שנותן המשתמש, אלא יעביר את הפקודה והארגומנטים למערכת בעזרת קריאה לפקודה ממשפחת הפקודות exec.

- במצב שקריאת מערכת נכשלה יש להדפיס הודעת שגיאה באופן הבא:

```
perror("{the name of the system call for example: fork} failed")
```

- מעבר לכך, אין להדפיס שגיאות נוספות מהתוכנית שלכם. מצב בו הפקודות עצמן ידפיסו שגיאה (כלומר לא עפ"י הגדרה שלכם) הוא תקין.
- ניתן להניח אורך פקודה מקסימלי של 100 תווים.
- ניתן להניח שיוכנסו בבדיקה לכל היותר 100 פקודות.
- את הprompt יש להדפיס באופן הבא:

```
printf("$ ");
fflush(stdout);
```

- ניתן להניח שלא יוכנסו רווחים בשמות תיקיות וקבצים.
- מומלץ להשתמש ב strtok עם דלימטר של רווח על מנת לפצל את הפקודה לרכיביה,

ניתן להניח שפיצול כנ"ל הוא תמיד נכון ולכן אין צורך לתמוך לדוג' ב "..“ echo (להעביר את כל מה שבגרשיים כפרמטר אחד אפילו אם יש בו רווחים)

- לקריאה ושינוי משתני סביבה אתם יכולים להשתמש ב getenv, setenv.

דוג' ריצה (אותה ריצה מתחיל משמאל ונגמר בימין):

```
root@DESKTOP-HKCE1KB:~/hw1_os# gcc myshell.c
root@DESKTOP-HKCE1KB:~/hw1_os# ls
0  a.out  my_command  myshell.c
root@DESKTOP-HKCE1KB:~/hw1_os# ls my_command/
gccfind.sh  tennis.sh
root@DESKTOP-HKCE1KB:~/hw1_os# tree 0
0
├── 1
│   ├── 3
│   │   ├── atexit_example_1.c
│   │   └── atexit_example_2.c
│   ├── 4
│   │   ├── execvp_wait_example.c
│   │   ├── fork_example_1.c
│   │   └── fork_example_2.c
│   ├── execl_example.c
│   └── execv_example.c
├── 2
│   ├── 5
│   │   ├── 6
│   │   │   ├── t2_2.c
│   │   │   └── t2_3.c
│   │   └── t2_1.c
│   └── fork_example_3.c
├── wait_example_1.c
├── wait_example_2.c
└── waitpid_example.c

6 directories, 14 files
root@DESKTOP-HKCE1KB:~/hw1_os# ./a.out /root/hw1_os/my_command
$ cat ./0/2/5/6/t2_2.c
#include <unistd.h>
#include <stdio.h>

int main(void)
{
    int status;
    char *argv[] = { "/bin/env", 0 };
    char *envp[] =
    {
        "HOME=/",
        "PATH=/bin:/usr/bin",
        "TZ=UTC0",
        "USER=beelzebub",
        "LOGNAME=tarzan",
        0
    };
    status=execve(argv[0], &argv[0], envp);
    printf("exec failed\n");
    return -1;
}
$ ls
0  a.out  my_command  myshell.c
$ ls -l
total 32
drwxr-xr-x 4 root root  4096 Apr  4 16:15 0
-rwxr-xr-x 1 root root 17456 Apr  4 16:23 a.out
drwxr-xr-x 2 root root  4096 Apr  4 16:13 my_command
-rw-r--r-- 1 root root  1639 Apr  4 15:44 myshell.c
$ gccfind.sh 0 child
```

```
$ tree 0
0
├── 1
│   ├── 3
│   │   ├── atexit_example_1.c
│   │   └── atexit_example_2.c
│   ├── 4
│   │   ├── execvp_wait_example.c
│   │   ├── fork_example_1.c
│   │   └── fork_example_2.c
│   ├── execl_example.c
│   └── execv_example.c
├── 2
│   ├── 5
│   │   ├── 6
│   │   │   ├── t2_2.c
│   │   │   └── t2_3.c
│   │   └── t2_1.c
│   └── fork_example_3.c
├── wait_example_1.c
├── wait_example_2.c
├── wait_example_2.out
├── waitpid_example.c
└── waitpid_example.out

6 directories, 16 files
$ gccfind.sh 0 parent -r
$ tree 0
0
├── 1
│   ├── 3
│   │   ├── atexit_example_1.c
│   │   └── atexit_example_2.c
│   ├── 4
│   │   ├── execvp_wait_example.c
│   │   ├── execvp_wait_example.out
│   │   ├── fork_example_1.c
│   │   └── fork_example_2.c
│   ├── execl_example.c
│   └── execv_example.c
├── 2
│   ├── 5
│   │   ├── 6
│   │   │   ├── t2_2.c
│   │   │   ├── t2_3.c
│   │   │   └── t2_3.out
│   │   ├── t2_1.c
│   │   └── t2_1.out
│   ├── fork_example_3.c
│   └── fork_example_3.out
├── wait_example_1.c
├── wait_example_2.c
├── waitpid_example.c
└── waitpid_example.out

6 directories, 19 files
```

```
$ tennis.sh
 Player 1: 50         Player 2: 50
 --------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       O       |       |
 |       |       #       |       |
 |       |       #       |       |
 --------------------------------
PLAYER 1 PICK A NUMBER:
PLAYER 2 PICK A NUMBER:
 Player 1: 0          Player 2: 0
 --------------------------------
 |       |       #       |       |
 |       |       #       |       |
 |       |       O       |       |
 |       |       #       |       |
 |       |       #       |       |
 --------------------------------
       Player 1 played: 50
       Player 2 played: 50


IT'S A DRAW !
$ cd 0
$ pwd
/root/hw1_os/0
$ cd ..
$ pwd
/root/hw1_os
$ history
24179 cat ./0/2/5/6/t2_2.c
24180 ls
24181 ls -l
24183 gccfind.sh 0 child
24221 tree 0
24222 gccfind.sh 0 parent -r
24313 tree 0
24330 tennis.sh
24177 cd 0
24331 pwd
24177 cd ..
24332 pwd
24177 history
$ cd 0/1
$ pwd
/root/hw1_os/0/1
$ history
24179 cat ./0/2/5/6/t2_2.c
24180 ls
24181 ls -l
24183 gccfind.sh 0 child
24221 tree 0
24222 gccfind.sh 0 parent -r
24313 tree 0
24330 tennis.sh
24177 cd 0
24331 pwd
24177 cd ..
24332 pwd
24177 history
24177 cd 0/1
24333 pwd
24177 history
$ exit
root@DESKTOP-HKCE1KB:~/hw1_os#
```