# מערכות הפעלה - תרגיל בית 1

אוניברסיטת חיפה

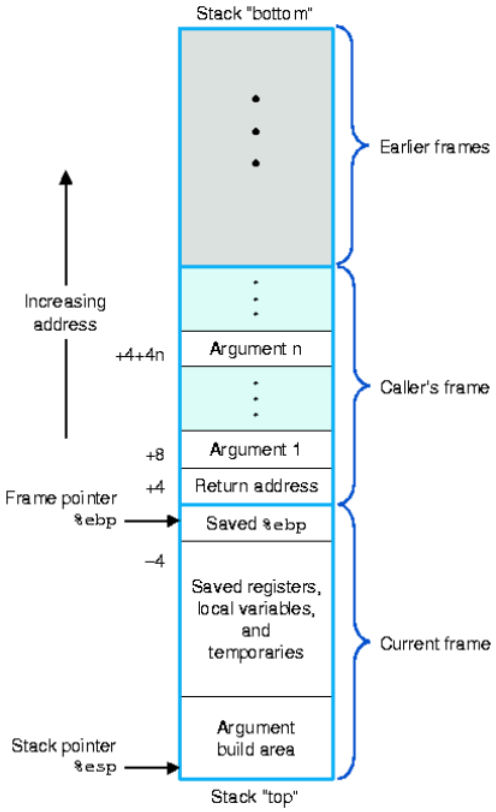
סמסטר אביב תשפ"ד

לתרגיל זה שני חלקים - תיאורטי ומעשי.

1. חלק תיאורטי (30 נקודות), בו יהיה עליכם לענות על מספר שאלות תיאורטיות.
2. חלק מעשי (70 נקודות), בו יהיה עליכם לממש shell פשוט.

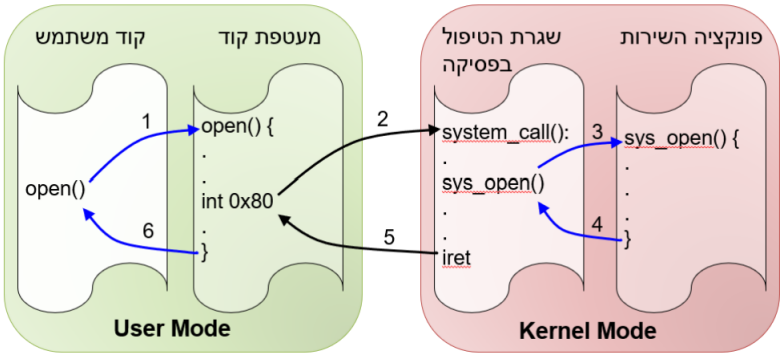
# חלק תיאורטי

1. הסבירו במשפט אחד מדוע בקונבנציית GCC, כאשר קוראים לפונקציה, שומרים במחסנית בסדר הפוך את הפרמטרים המועברים לה, כמתואר באיור 1.




איור 1: העברת פרמטרים למחסנית, תחילה push ל-$n$ Argument, עד ל-1 Argument בסוף.

2. הסבירו במשפט אחד מדוע לא ניתן להעביר פרמטרים דרך המחסנית בזמן ביצוע קריאת מערכת, system call, כאשר עוברים מ-user mode ל-kernel mode. היעזרו באיור 2.




איור 2: פונקציית המעטפת (באיזור הירוק) אינה יכולה להעביר פרמטרים דרך המחסנית לשגרת הטיפול בפסיקה (באיזור האדום).

3. ציינו את כל הפלטים האפשריים (stdout) של קטע הקוד באיור 3. (**נמקו**)

```
pid_t pid = fork();
if (pid < 0) exit(1);
else if (pid > 0) {
        printf("%d", getpid());
        exit(0);
}
else {
        char *argv[] = {"sleep", "1", NULL};
        execv("/bin/sleep", argv);
        printf("%d", getpid());
}
```

איור 3: קטע קוד קצר. הניחו כי $\text{pid}\,(\text{child}) = 13$, $\text{pid}\,(\text{parent}) = 8$.

4. ציינו את כל הפלטים האפשריים (stdout) של קטע הקוד באיור 4. (**נמקו**)

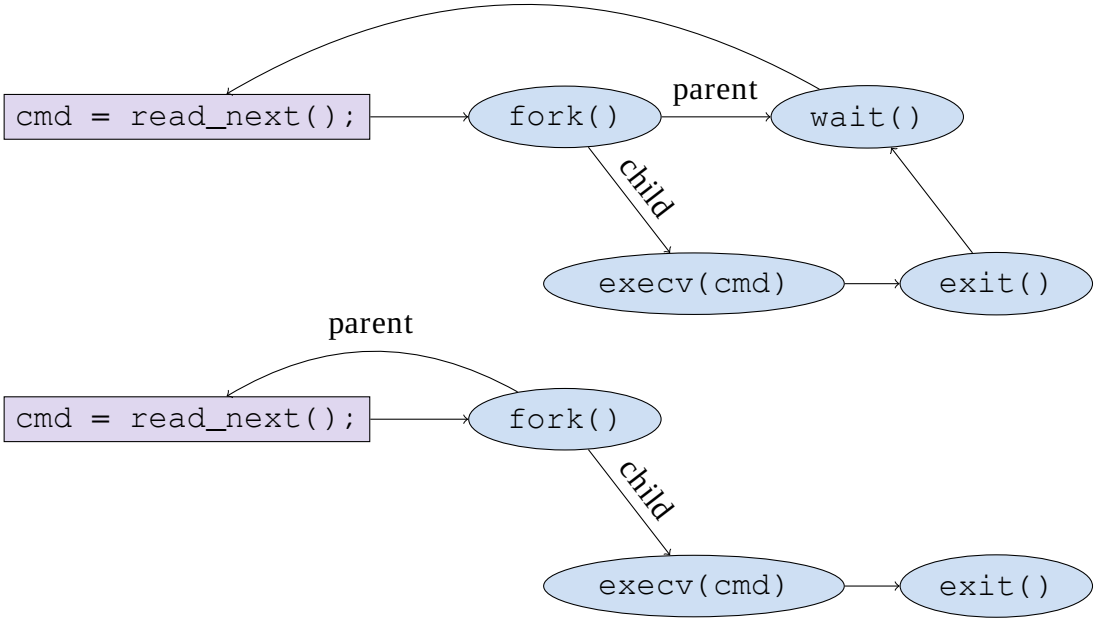
```
int value = 0;
if (fork() != 0) {
        wait(&value);
        value = WEXITSTATUS(value);
        value += 3;
}
printf("%d\n", value);
value += 4;
exit(value);
```

איור 4: קטע קוד קצר.

# חלק מעשי

## הנחיות

בחלק זה תממשו תוכנית shell פשוטה, שתאפשר לחמשתמש להריץ פקודות בחזית וברקע (ראו איור 5).




איור 5: מלמעלה למטה: הרצה בחזית, הרצה ברקע.

- התוכנית תדפיס למסך את המחרוזת "`my-shell>`" לפני קבלת הפקודה הבאה מהמשתמש.
- לאחר שהמשתמש הקליד את הפקודה (למשל "`mkdir newfolder`"), התוכנית תריץ את הפקודה.
- אם הפרמטר האחרון בפקודה שהוקלדה היה התו "&", התוכנית תריץ את הפקודה ברקע, ואחרת בחזית.
- התוכנית תסיים את ריצה כאשר המשתמש יקליד "`exit`".

⊕ בנוסף, עליכם לממש פקודה נוספת בשם "history", בה יודפסו למסך כל הפקודות שהמשתמש הקליד בעבר.

⊗ כל פקודה בשורה נפרדת, יחד עם המספר הסידורי שלה.

⊗ הפקודות יודפסו בסדר יורד מזמן ביצוע הפקודה (הפקודה האחרונה תודפסה בשורה הראשונה).

⊗ דוגמת הרצה:

| | | | |
|---|---|---|---|
| date | | 5 | history |
| mkdir dir1 | | 4 | echo hi & |
| mkdir dir2 | ⇒ | 3 | mkdir dir2 |
| echo hi & | | 2 | mkdir dir1 |
| history | | 1 | date |

**הערות.**

1. הרצה לדוגמא בלינק.
2. מסופק לכם קובץ שלד של התוכנית - myshell.c.
3. השתמשו ב-strtok כדי להפריד את הפקודה לחלקיה השונים, ע"י שימוש ברווח (whitespace) כמפריד (delimiter) בין חלקי הפקודה.
4. **לא דרוש מכם לממש בעצמכם פקודות כגון** `ls` **או** `mkdir`.

   ⊕ עליכם להשתמש ב-`execvp` כדי להריץ את הפקודות שקיבלתם מהמשתמש (דוגמא לשימוש ב-`execvp`).
5. על `history` לכלול גם פקודות שכשלו ולא התבצעו, ואת `history` בעצמה.
6. שימו לב לפורמט ההדפסה של ההיסטוריה, כמפורט לעיל.
7. **לא דרוש מכם לתמוך בפונקציות נוספות של ה-shell שלא ניתן לבצע ע"י** `exec`**, כמו pipes או cd.**

- ⊕ כדי לדעת האם פקודה x ניתנת לביצוע ע"י exec, הריצו ב-shell "which x".
- ⊕ הפקודה ניתנת לביצוע אמ"מ קיבלתם את המסלול לקובץ שנמצא ב-shell (הריצו למשל which history ב-shell רגיל, ותראו את התוצאה).

8. מספיק שהרישא של history תהיה חוקית כדי שהפקודה תתבצע, אך בהיסטוריה היא תישמר בדיוק כפי שהמשתמש הקליד.
   - ⊕ למשל, עבור הקלט history12, הפקודה תתבצע ותישמר בהיסטוריה כ-history12.
9. בכל מקרה של שגיאה השתמשו בפקודה perror("error").

## קומפילציה והרצה

באופן דומה לתרגיל בית 0, הדרו את התרגיל באמצעות הפקודה

```
gcc -Werror -std=c99 myshell.c -o myshell
```

הריצו את התוכנית באמצעות

```
./myshell
```

## הגשה

הגשה במודל, לפי הפורמט הבא:

1. עליכם ליצור קובץ zip (השתמשו ב-zip או gzip בלבד) בשם hw1_id1_id2.zip, כאשר id1, id2 הם מספרי תעודות הזהות של המגישים.
2. קובץ ה-zip מכיל **אך ורק** את הקבצים הבאים, **ללא תתי-ספריות**.
   - ⊕ myshell.c
   - ⊕ dry.pdf, שמכיל את התשובות לחלק התיאורטי - **קובץ pdf בלבד**.

⊕ submitters.txt, שמכיל את מספרי הזהות והשמות של מגישי התרגיל, מופרדים ע"י פסיק. למשל:

```
Bill Gates,bill@microsoft.com,123456789
Linus Torvalds,linus@gmail.com,234567890
```

3. צרו את קובץ ה-zip באמצעות הפקודה

```
zip hw1_id1_id2.zip myshell.c dry.pdf submitters.txt
```