תרגיל בית 1

תיאור התרגיל

לתרגיל זה שני חלקים, חלק מעשי וחלק תיאורטי.

חלק תיאורטי (30% ניקוד)

בחלק זה יהיה עליכם לענות על מספר שאלות תיאורטיות.

חלק מעשי (70% ניקוד)

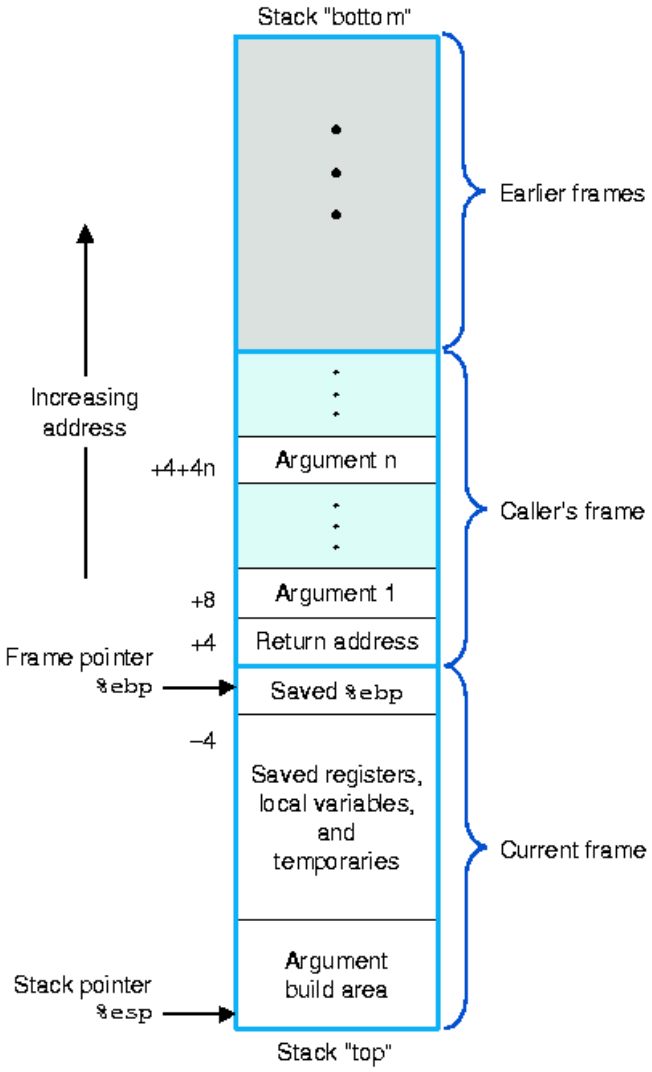
בחלק זה יהיה עליכם לממש שתי תוכניות.

1. בתוכנית הראשונה תממשו shell פשוט (תוכנית שורת פקודה, בדומה ל- shell הקיים ב- Linux).

חלק תיאורטי

שאלה 1

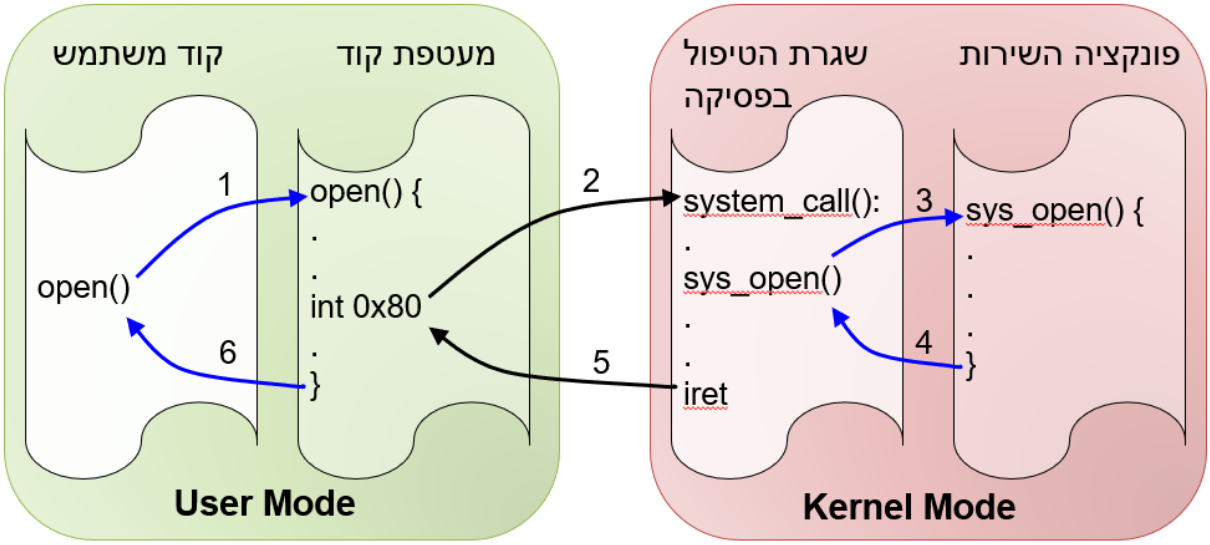
הסבירו **<u>במשפט אחד בלבד</u>** מדוע בקונבנציית GCC לקריאה לפונקציה, שומרים בסדר הפוך (במחסנית) את הפרמטרים המועברים לפונקצייה הנקראת:



כלומר, הסבירו מדוע תחילה מבצעים push לפרמטר ה- n (אחרון), ורק לבסוף מבצעים push לפרמטר ה- 1 (ראשון).

שאלה 2

הסבירו **<u>במשפט אחד בלבד</u>** מדוע לא ניתן להעביר פרמטרים דרך המחסנית, בזמן ביצוע קריאת מערכת (System call), כאשר עוברים מה- User mode ל- Kernel mode. כלומר, הסבירו מדוע "פונקציית המעטפת" (באיזור הירוק) לא יכולה להעביר פרמטרים דרך המחסנית לשגרת הטיפול בפסיקה (באיזור האדום), כפי שמתואר בתרשים הבא:



שאלה 3

ציינו את כל הפלטים האפשריים (למסך) של קטע הקוד הבא: **(נמקו!)**

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

<u>ניתן להניח כי:</u>

```
pid(father) = 8
pid(son) = 13
```

שאלה 4

ציינו את כל הפלטים האפשריים (למסך) של קטע הקוד הבא: **(נמקו!)**

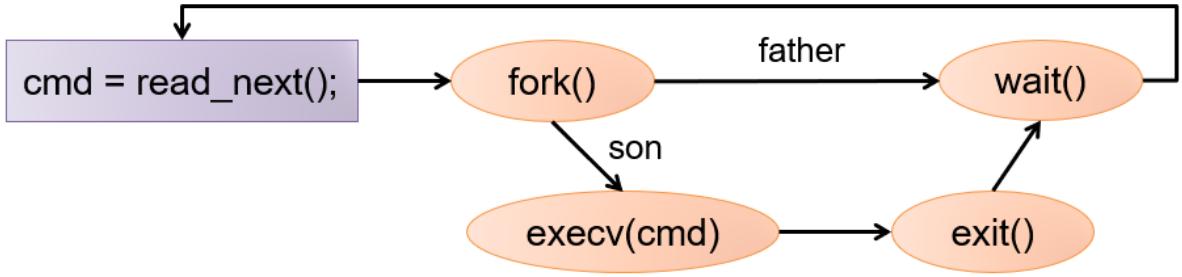
```
int value = 0;

if (fork() != 0)

{

       wait(&value);

       value = WEXITSTATUS(value);

       value += 3;

}

printf("%d\n", value);

value += 4;

exit(value);
```

חלק מעשי

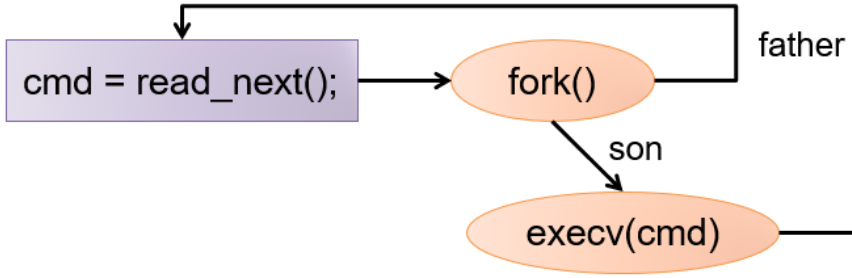
מימוש Shell

עליכם לממש תוכנית Shell פשוטה, אשר תאפשר למשתמש להריץ פקודות בחזית וברקע.

- הרצה בחזית:



- הרצה ברקע:



התוכנית תדפיס למסך את המחרוזת " my-shell>" לפני קבלת הפקודה הבאה מהמשתמש. לאחר שהמשתמש הקליד את פקודתו (למשל "mkdir newfolder"), התוכנית תריץ את הפקודה. אם הפרמטר האחרון שהמתמש הקליד (כחלק מהפקודה) היה התו "&" אז התוכנית תריץ את הפקודה ברקע, אחרת, התוכנית תריץ את הפקודה בחזית. התוכנית תסיים את ריצתה כאשר המשתמש יקליד "exit".

כמו כן, עליכם לממש פקודה נוספת בשם "history". כאשר המשתמש יקליד את הפקודה "history", יודפסו למסך כל הפקודות שהמשתמש הקליד בעבר, כל אחת בשורה נפרדת, בסדר יורד מבחינת זמן ביצוע הפקודה (כלומר, הפקודה האחרונה שהורצה, תודפס בשורה הראשונה). למשל, אם המשתמש הריץ קודם כל את הפקודה ls, לאחר מכן את הפקודה who, ולבסוף את הפקודה date, אז מה שיודפס למסך יהיה:

```
date

who

ls
```

<u>מספר הערות:</u>

1. **סרטון אחד שווה אלף מילים, ולכן מצורף לינק ל- YouTube בו תוכלו לראות הרצה לדוגמה.**

2. מסופק לכם קובץ myshell.c עם שלד התוכנית שעליכם לממש.
3. השתמשו ב- strtok כדי להפריד את הפקודה לחלקיה, ע"י שימוש בתו רווח (whitespace) כמפריד בין חלקי הפקודה (delimiter).
4. **שימו לב:** במידה וזה לא היה ברור – אין עליכם לממש פקודות Linux Shell כגון ls או mkdir, כל שעליכם לעשות הוא להשתמש במתודה execvp כדי להריץ פקודות (כגון הנ"ל) שהמשתמש הקליד. דוגמה לשימוש ב-execvp ניתן למצוא כאן.

קומפילציה והרצה

תוכנית Shell

בדומה למפורט בתרגיל בית 0, כדי להדר את התוכנית, הריצו את הפקודה הבאה:

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

בכדי להריץ התוכנית, הריצו את הפקודה:

```
./myshell
```

הגשה

ההגשה הינה אלקטרונית דרך Moodle. עקבו אחר השלבים הבאים:

1. עליכם ליצור קובץ zip (השתמשו ב-zip או gzip בלבד) בשם hw1_id1_id2 כאשר id1,id2 מייצגים את מספרי תעודות הזהות של המגישים.
2. תכולת קובץ ה zip צריכה להיות התכולה הבאה (ללא תתי ספריות!):
   - myshell.c
   - dry.pdf (עם התשובות של החלק התיאורטי – **אך ורק קובץ PDF**).
   - קובץ בשם submitters.txt שמכיל את מספרי הזהות והשמות של מגישי התרגיל מופרדים על ידי פסיק במבנה הבא (לדוגמה):

```
Bill Gates,bill@microsoft.com,123456789

Linus Torvalds,linus@gmail.com,234567890
```

3. את קובץ ה- zip יש ליצור ע"י הרצת הפקודה הבאה:

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

4. הגישו את קובץ ה- zip דרך Moodle.