

נספח הסבר - פרויקט Mini-Bash Shell (מטלה 3)

1. תיאור כללי

בפרויקט זה מימשי מפרש פקודות (Shell) בסיסי בשפת C. התוכנית מאפשרת למשתמש להקליד פקודות חיצוניות (כמו `ls`, `pwd`) ופקודות פנימיות (`cd`, `exit`), ומנהלת את הרצתן מול ה-Kernel של לינוקס.

2. מבנה הלוגיקה המרכזית (The Shell Loop)

התוכנית רצה בלולאה אינסופית המבצעת את השלבים הבאים:

- **הצגת Prompt:** הדפסת המחרוזת `$mini-bash` למסך כדי לסמן למשתמש שהמערכת מוכנה לקלט.
- **קריאת פקודה (Read):** שימוש בפונקציה `fgets` לקבלת שורת הטקסט מהמשתמש.
- **ניתוח (Parse):** פירוק השורה למילים (Tokens) באמצעות הפונקציה `strtok`. הפירוק מתבצע על בסיס רווחים וטאבים.
- **זיהוי וביצוע:** בדיקה האם המשתמש הקיש פקודה פנימית או חיצונית, והפעלת המנגנון המתאים.

3. קריאות מערכת (System Calls) בשימוש

כדי לעמוד בדרישות התרגיל, השתמשתי בקריאות המערכת הבאות:

- **`fork()`:** ליצירת תהליך בן (Child Process) חדש שבו תבוצע הפקודה החיצונית, בזמן שתהליך האב (ה-Shell) ממתיין.
- **`execvp()`:** להחלפת התוכן של תהליך הבן בקוד של הפקודה המבוקשת. הפונקציה מחפשת את הפקודה בנתיבי המערכת ובתיקיית הבית.
- **`waitpid()`:** סנכרון בין האב לבן. האב ממתיין לסיום ריצת הבן ומקבל את ה-Return Code (ערך החזרה) שלו.
- **`chdir()`:** מימוש פקודת ה-`cd`. מכיוון ששינוי תיקייה משפיע על התהליך עצמו, הוא חייב להתבצע ישירות בתהליך האב.
- **`perror()`:** הדפסת הודעות שגיאה מפורטות מה-Kernel במקרה של כשל ב-`fork` או ב-`exec`.

4. טיפול בשגיאות ויעילות

- **יעילות:** השימוש ב-`strtok` מאפשר לפרק את המחרוזת ללא הקצאות זיכרון נוספות, מה שחוסך במשאבים.
- **שגיאות:** במקרה שהפקודה אינה קיימת באף אחד מהנתיבים, התוכנית מדפיסה הודעת שגיאה בפורמט: `[command_name]: Unknown Command`.