

נספח הסבר - פרויקט Mini-Bash Shell (מטרה)

(3)

1. תיאור כללי

בפרויקט זה מימושי מפרש פקודות (Shell) בסיסי בשפת C. התוכנית מאפשרת למשתמש להקליד פקודות חיצונית (כמו `cd`, `ls`, `pwd`) ופקודות פנימיות (כמו `exit`) ומנהלת את הרצtan מול ה-Kernel של Linux.

2. מבנה הלוגיקה המרכזית (The Shell Loop)

התוכנית רצה בלולאה אינסופית המבצעת את השלבים הבאים:

- **הציגת 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 למיללים (Tokens) באמצעות הפונקציה `strtok`. הפירוק מתבצע על בסיס רווחים וטאבילים.
- **דיהוי וביצוע:** בדיקה האם המשתמש הקיש פקודה פנימית או חיצונית, והפעלת המנגנון המתאים.

3. קראות מערכת (System Calls) בשימוש

כדי לעמוד בדרישות התרגיל, השתמשתי בקראות המערכת הבאות:

- **fork()**: ליצור תהליך בן (Child Process) חדש שבו תבוצע הפקודה החיצונית, בזמן שתתהליך האב (the-Shell) ממתין.
- **execvp()**: להחלפת התוכן של תהליך הבן בקוד של הפקודה המבוקשת. הפונקציה מוחפשת את הפקודה בתארכי המערכת ובתיקיית הבית.
- **waitpid()**: סyncron בין האב לבן. האב ממתין לסיום ריצת הבן ומקבל את ה-Return Code (ערך החזרה) שלו.
- **chdir()**: שימוש פקודת `cd`. מכיוון ששינוי תיקייה משפיע על התהליך עצמו, הוא חייב להתבצע ישירות בתהller 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 נוספת, מה שחוסך במשאבים.
- **שגיאות:** במקרה שהפקודה אינה קיימת אף אחד מהנתיבים, התוכנית מדפסה הודעה שגיאה בפורמט: [command_name : Unknown Command]