**רטוב 1 מערכות הפעלה**

מגישים:   
עדי יצחקי   
אלישע פריד

אפיון התכנית:

יצרנו תוכנית שהיא בעצם קליפה חיצונית שמדברת עם הbash באמצעות פקודות שמכניס הuser ושנשלחות באמצעות גוף התוכנית שמעבד את הקלט למודולים ופונקציות שמטפלות בכל פקודה לפי הסוג שלה.

דיאגרמת בלוקים:

**commands.cpp**

המודול שעושה את כל העבודה השחורה ולוקח את כל סוגי הפקודות שמקבל מהsmash ומבצע כל אחת לפי דרך הביצוע שלה.

**signals.cpp**

מודול שמטפל בקריאה לאיתותים ע"י הuser באמצעות ctrl-c ctrl-z  
וכן בפונקציה חיצונית שקוראת לפונקציות שמטפלות בכל איתות ספציפי.

**smash.cpp**

המודול שהוא גוף הsmash (main( אשר מקבל קלט ומטפל בו ע"י שליחה למודולים האחרים

מנשק בין המודולים:

הקשר בין הsmash לcommands כדי להעביר את הקלט שהוכנס ע"י המשתמש ולהעבירו בצורה מפורקת ומכוונת לפונקציות בcommands

הפונקציות בcommands מחזירות לsmash מידע אם הצליחו או נכשלו וכך הsmash יעביר לטיפול אחר, המשך קבלת קלט או להודעת שגיאה במידה והקלט לא חוקי.   
המידע שעובר בין הsignals לשאר המודולים נוגע בתהליך שכרגע רץ וכן עדכון משתנים גלובלים שנעשה בהם שימוש בכל הפונקציות כמו וקטור jobs וקטור waiting ועוד שנרחיב בהמשך.  
זרימת המידע:

* הגדרת משתנים גלובלים שישמשו אותנו במהלך ריצת התכנית
* קביעת handlers שמטפל ב ctrl-c ו ctrl-z
* כניסה ללולאת קלט אינסופית עד לסיום התכנית ושליחת הקלט לפונקציות טיפול בcommands.cpp:

1. פונקציה לטיפול בפקודות מסובכות (המענה המתקבל הוא אי טיפול)
2. פונקציה לטיפול בפקודה חיצונית רגילה (לא background), הפונקציה מפצלת לתהליך בן חדש אשר נשלח לטפל בפקודה החיצונית בעוד האב ממתין לביצוע באמצעות waitpid.
3. פונקציה לטיפול בפקודה חיצונית עם & אשר מסמל הרצה ב background, לצורך כך הפונקציה מפצלת לתהליך בן חדש אשר נשלח לבצע את הפקודה אך האבא לא מחכה לביצועו וממשיך ללואה לקבל קלט חדש. כמו כן נוצר Job חדש(עם כל המאפיינים שלו) המתאר את תהליך הבן שמתווסף לרשימת הjobs כדי לאפשר מעקב אחר תהליכים שרצים ברקע(וכך אפשר להפסיק אותם לשלוח להם סיגנלים וכו').
4. פונקציה לטיפול בפקודות רגילות כגון pwd fg bg וכו' אשר משתמשת בעת הצורך במשתנים הגלובלים ויכול להעביר פקודות להמתנה או לפעולה וכן הלאה.

מבני נתונים:

Job-לצורך טיפול בפקודות ושמירה שלהן יש צורך ב class Job שמכיל שם פקודה את הpid שלה וכן את הזמן שבו היא נכנסה לוקטור הjobs ונוצרה למעשה.

vector <Job> jobs- וקטור של תהליכים המכיל את כל התהליכים שרצים כרגע ברקע ו\או מושהים. בחרנו בוקטור כי זה מבנה נתונים דינאמי שמאפשר לעבוד איתו בצורה מגוונת גם בתור רשימה וגם בתור מחסנית.

vector <Job> waiting- וקטור של עבודות שהושהו. הבחירה בוקטור גם כן כנ"ל מטעמי דינאמיות ונוחות.

runningPid/mainPid- משתנים גלובלים ששמרו את הpid גם של התכנית שרצה כרגע על המעבד וגם את הpid של הsmash עצמו, לצורך השוואה לכל אורך התכנית, מכיוון שהעדכון עבור התכניות שרצות נעשה לכל אורך התכנית ומטעמי נוחות בחרנו במשתנה גלובלי.

vector <string> history-שומרים בתוך הוקטור את הפקודות שהוכנסו ע"י המשתמש לפי הסדר, וקטור מאפשר לעשות זאת בצורה דינאמית ונוחה בדומה לרשימה.