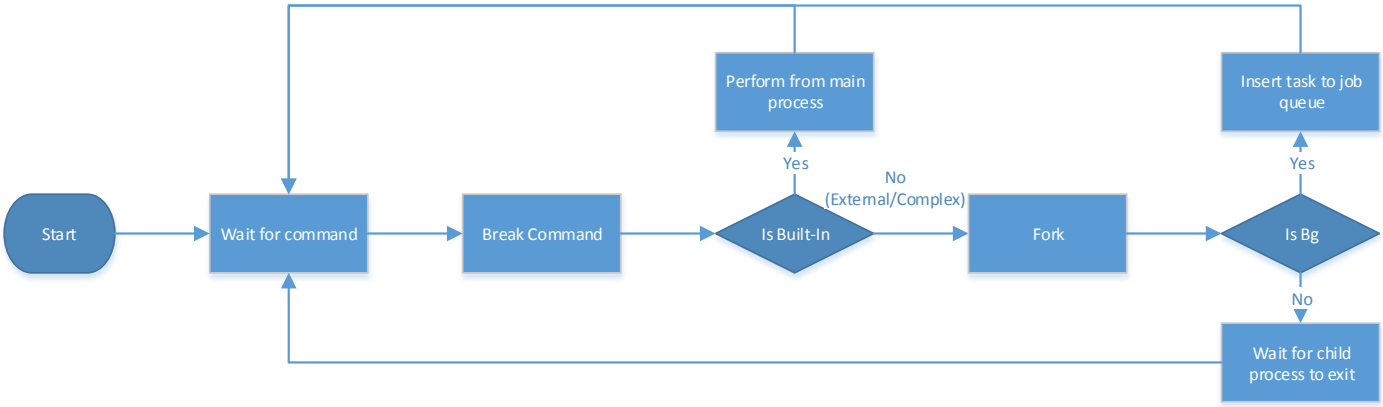
Operating Systems – Wet 1

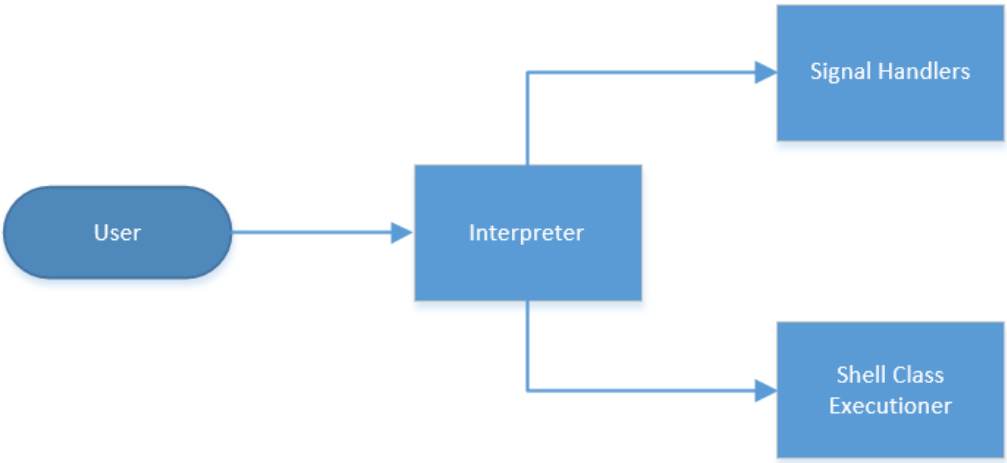
מגישים:

- דרור קבלי 311177109
- רון לוי: 201121613

אלגוריתם הריצה הכללי של ה-shell:



והדיאגרמה הכללית של ה-Shell היא:



1. **הסבר התהליך הראשי:**

התהליך מחכה לתגובה מהמשתמש (הכנסת שורה). לאחר הכנסת השורה, יש קריאה ל-Interpreter שמפענח איזו סוג פקודה.

אם הפקודה הינה פקודה Built-in, יש קריאה מיידית למחלקת ה-Shell שמבצעת את פקודת ה-Builtin, ולאחר מכן חוזרת.

אם הפקודה אינה Built-In, ה-Interpreter בודק אם הפקודה צריכה לרוץ ברקע (ע"י &) או לתפוס את החזית. לאחר הבדיקה, תהליך האב מבצע fork. לאחר ביצוע ה-fork, במקרה שהתהליך מורץ ברקע אין צורך לחכות לתהליך הבן שיסיים ואפשר לחזור, ובמקרה שלא רץ ברקע יש לחכות. לאחר סיום ריצת תהליך ורק לאחר מכן לחזור.

2. הסבר דיאגרמת הבלוקים:

ה-Interpreter מחכה לפעולה של המשתמש. כמו כן, ה-Interpreter מנתב את הפקודה שפענח מהמשתמש אל מחלקת ה-Shell שמבצעת בהתאם את הפעולה. הממשק בין ה-Interpreter ל-Shell נעשה על ידי הפעלת ה-Shell בפקודות הממשק של ה-Interpreter עם המשתמש.

ה-Signal Handlers נשלטים גם על ידי ה-Shell (אלה שמטפלים בסיגנלים שמקבל תהליך האב) וגם על ידי ה-Interpreter (אלה שמטפלים בסיגנלים שמקבל תהליך הבן)

כל פקודה שמוזנת על ידי המשתמש נשמרת במבנה נתונים ששומר את היסטוריית ההקלדות של המשתמש.

כל פקודה שצריכה לבצע fork שומרת את המידע שלה במבנה נתונים של ה-Shell, עד שהיא יוצאת מהמערכת, כולל הפקודה שרצה בחזית.

כל פקודה שהופסקה (SIGTSTP) על ידי המשתמש נשמרת במבנה נתונים נפרד של ה-Shell עד שיש בקשה להחזיר אותה לביצוע (fg ו-bg).

בסיום/הפסקת ריצה כלשהי יש קריאה לפונקציות מחלקה של ה-Shell שמשנות את מצבה בהתאם לפקודה שיצאה/הפסיקה, כלומר, הוצאה ממבני הנתונים של ה-Shell או הכנסה לרשימת הפקודות שהופסקו.

3. הסבר מחלקת ה-Shell

מחלקת ה-Shell תוכננה כמחלקת Singleton על מנת להבטיח יחידות של מבני הנתונים של המערכת ועל מנת לסוכך על המימוש הפנימי ומניפולצייה לא מבוקרת על מבני הנתונים של ה-Shell.

a. הסבר מבני הנתונים של ה-Shell

i. קונטיינר Jobs – קונטיינר vector (STL) ששומר מידע על כל אחד מהתהליכים שנמצאים כעת במערכת. מידע זה מאורגן בתת-מחלקה שנקראת job והנתונים שהיא מחזיקה אלו הם:
- ה-PID של התהליך
- מספר התכנית (task ID) שניתן לה ע"י ה-Shell
- מחרוזת שורת הפקודה עצמה
- סטטוס הריצה של התהליך (פעיל/הופסק)
- זמן יצירת התהליך

(לא נרחיב כאן יותר על מחלקת job מעבר לעובדה שהיא יכולה לבצע השמה, יש לה copy constructor על מנת לתמוך בפעולות השמה בסיסיות ויש לה ממשק שמחזיר את כל אחד מהנתונים שהיא שומרת)

ii. קונטיינר history – קונטייר deque (STL) שמחזיק מחרוזות על רצף הפקודות שהוכנסו למערכת. הקונטיינר יחזיק לכל היותר היסטוריה של 50 פקודות ולאחר מכן ימחק את הפקודה הכי ישנה כדי להכניס עוד שורות.

iii. קונטיינר stopped_procs – קונטיינר deque (STL) שמחזיק מבני נתונים של job (כפי שהוסבר ב- (i)). מחזיק את כל הפקודות שרצו ברקע, קיבלו SIGTSTP ועדיין לא חזרו לריצה. בעת החזרת תהליך כלשהו מהקונטייר לריצה, יש להוציא את התהליך מהקונטיינר.

הסיבות לבחירת מבני הנתונים שנבחרו לעיל יפורטו כדלהלן:

1. קונטיינר ה-vector תומך בפעולות של דחיפת איבר לסוף הווקטור והוצאת איבר מסוף התור, ולכן אפשר להשתמש בו כמחסנית. כמו כן, בניגוד ל-std::stack ניתן לגשת ל-vector בצורה אקראית כמו למערך.
2. קונטיינר ה-deque שנבחר עבור רגיסטר ההיסטוריה יכול לשמש בתור queue שכן הוא תומך בפעולות push_front ו-pop_back ומאפשר הוצאת נתונים מהירה מסוף התור והכנסה מהירה לתחילתו מה שעוזר לנו במקרה שרגיסטר ההיסטוריה מתמלא וצריך להוציא ממנו את הפקודות הכי ישנות (מסוף התור) ובמקומו להכניס את הפקודות החדשות בקדמת התור.
3. קונטיינר ה-deque נבחר עבור רגיסטר ה-stop procs על מנת לאפשר הוצאה מהירה מקדמת התור (מה ש-vector לא תומך בו בכלל), דבר שמאפשר לנו להוציא במהירות איבר במקרה ש-bg לא מקבל שום ארגומנט אחר.

<u>כל 3 הקונטיינרים לעיל נבחרו תחת הדרישה שלנו לגמישות פיתוח הקוד ותמיכה באלגוריתמי STL הבסיסיים כמו find_if, remove_if, rotate ו-sort.</u>

כל שימוש באלגוריתם דורש שימוש ב-functor מיוחד (אין תמיכה בלאמבדות בגלל הקומפיילר הישן שעל השרת) שכמובן מצורף לקוד ומתועד בהתאם.

a. **<u>משתנים גלובליים:</u>**

- נציין בנוסף שמחלקת ה-Shell מחזיקה את ה-pid של התהליך שכרגע רץ ברקע, את ה-pid של ה-smash, את ה-paths לתיקיית העבודה הנוכחית, זאת של תיקיית האב וזאת של תיקיית ה-home, משתנה סטטי שקובע את ה-task ID של ה-job הבא שתיכנס למערכת ואת הדגל שמהווה lock כאשר יש צורך להריץ תהליך בחזית.

<u>(*) הערה: מאחר ומחלקת ה-shell מתוכננת בתור singleton, אין יותר מ-instance יחיד של כל משתנה ואפשר להתייחס אליהם בתור משתנים גלובליים.</u>

(**) התיעוד לפונקציות המחלקה של מחלקת ה-Shell מפורטות בהתאם בתוך קובץ ה-shell.cpp.

(***) כל פונקציות ה-Built in של ה-smash ממומשות כפונקציות מחלקה של Shell.