REPUBLIKA E SHQIPERISË

UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT
DEPARTAMENTI I INXHINIERISË INFORMATIKE

DETYRË KURSI

Lënda: Sisteme Operative

Dega: Inxhinieri Informatike

Grupi: III-B

Punoi: Piro Gjikdhima **Pranoi: Dr. Dorian Minarolli**

Viti Akademik 2024-2025

Përmbajtja

1 Hyrje

Ky projekt paraqet dizenjimin dhe implementimin e një shell-i të thjeshtë në gjuhën programuese C. Shell-i i zhvilluar ofron funksionalitete bazë që lejojnë ndërveprim të përdoruesit me sistemin operativ përmes komandave të ndryshme. Ky dokument përmban detaje mbi kërkesat e punës, implementimin, dhe rezultatet e arritura gjatë realizimit të projektit.

2 Kërkesat e Punës

Ky projekt synon të ndërtimin e një shell të thjeshtë në gjuhën programuese C që ofron funksionalitete bazë të një shell-i Unix. Shell-i duhet të plotësojë kërkesat e mëposhtme:

1. Prompt Interaktiv:
    - Shfaq një prompt me simbolin "$>".
    - Pret input nga përdoruesi dhe e përpunon atë.
2. Ekzekutimi i Komandave të Thjeshta:
    - Ekzekutimi i komandave bazë të sistemit operativ si "ls", "pwd", etj.
3. Ridrejtimi i Input/Output:
    - Ridrejtim i input-it nga një file me formatin: komanda < file.
    - Ridrejtim i output-it në një file me formatin: komanda > file.
4. Mbështetje për Pipe:
    - Lejon përdorimin e simbolit "|" për të lidhur dy komanda në formatin: komanda1 | komanda2.
    - Output-i i komandës së parë përdoret si input për komandën e dytë.
5. Komanda në Background:
    - Lejon ekzekutimin e komandave në background duke përdorur simbolin "&" me formatin: komanda &.

3 Implementimi

3.1 Përshkrimi i programit kryesor

3.1.1 Programi main.c

Ky program implementon nje shellin e thjeshtë që pranon dhe ekzekuton komanda nga perdoruesi. Funksionalitetet kryesore janë:

1. **Prompt Interaktiv**
    - Shfaq një prompt blu "$>"
    - Pret për input nga përdoruesi
2. **Llojet e Komandave që Mbështet**:
    - cd: Për ndryshimin e direktorisë
    - Komanda background (me &)
    - Komanda Pipe (me |)
    - Komanda redirection (me > ose <)
    - Komanda të thjeshta ekzekutimi
    - exit për mbylljen e shell-it
3. **Veçoritë**:
    - Madhësia maksimale e komandës: 1024 karaktere
    - Përdor ngjyra ANSI për output
    - Heq hapësirat e panevojshme nga komandat
    - Trajton karaktere special (&, |, >, <)
4. **Trajtimi i Gabimeve**:
    - Validim për komandën 'cd'
    - Mesazhe gabimi me ngjyrë të kuqe
    - Flush i stdout për output të menjëhershëm

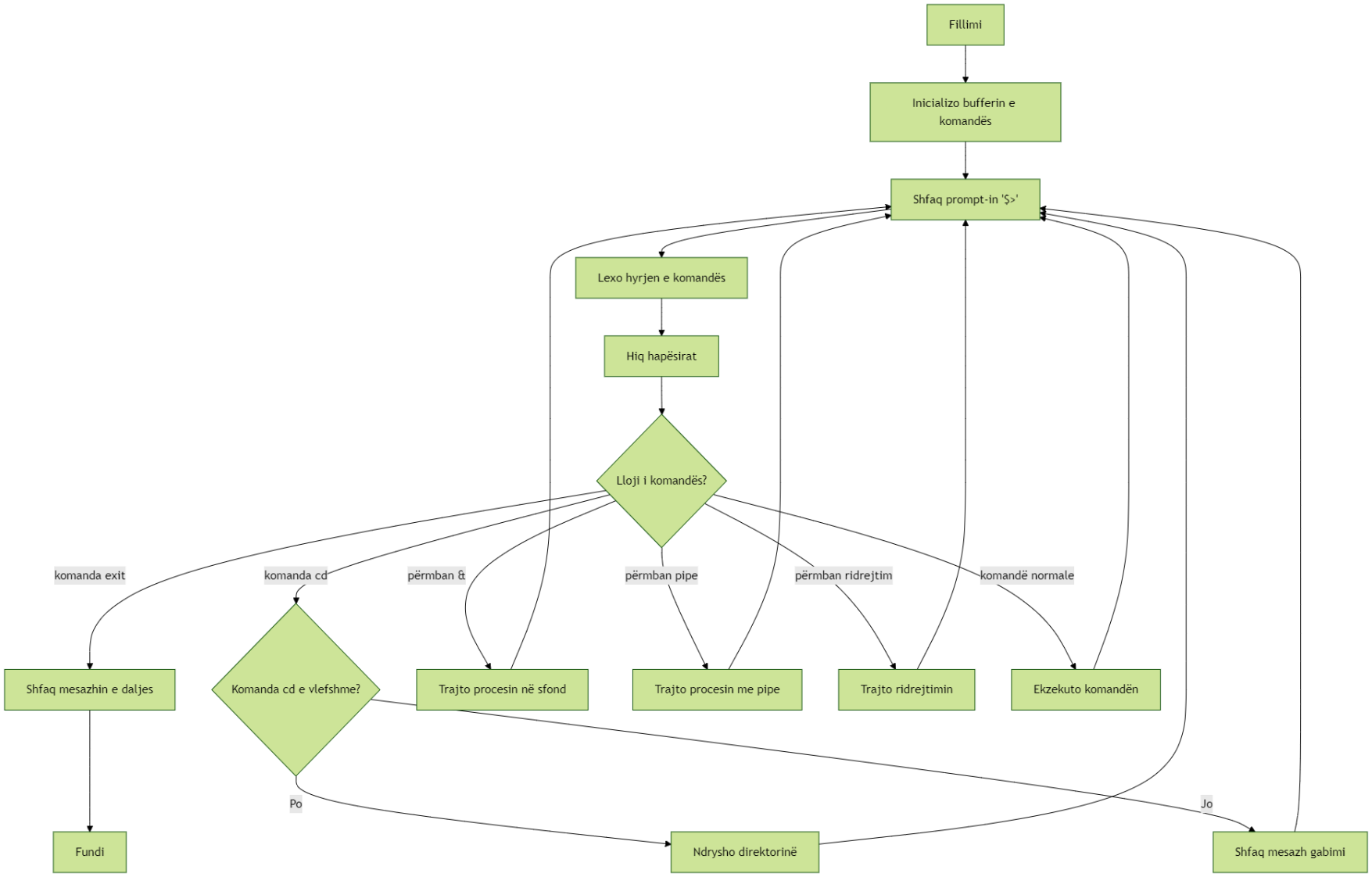
Përdorimi

1. Programi fillon dhe shfaq promptin $>
2. Përdoruesi fut një komandë
3. Komanda procesohet bazuar në përmbajtjen e saj
4. Rezultati shfaqet dhe prompti shfaqet përsëri
5. Procesi vazhdon derisa të jepet komanda exit (ose CTRL-c)

3.1.2 Pseudokodi

```
DEFINE MAX_COMMAND_LENGTH = 1024

FUNCTION main():
   DECLARE command as char array[MAX_COMMAND_LENGTH]

   WHILE true DO:
      SET_COLOR to blue
      PRINT "$> "
      FLUSH stdout

      command = READ_LINE()
      REMOVE newline from command
      SET_COLOR to green

      remove_spaces(command)

      IF command equals "exit" THEN
         SET_COLOR to green
         PRINT "Exiting ..."
         BREAK
      ELSE IF command starts with "cd" THEN
         IF command[2] is empty OR contains space THEN
            CALL cd with command[3:]
         ELSE
            PRINT "Invalid command: path expected after 'cd'" in red
         END IF
      ELSE IF command contains "&" THEN
         CALL background with command
      ELSE IF command contains "|" THEN
         CALL pipe_process with command
      ELSE IF command contains ">" OR "<" THEN
         CALL redirect_process with command
      ELSE
         CALL execute with command
      END IF
   END WHILE
   RETURN 0
END FUNCTION
```

3.1.3 Bllokskema



3.2 Përshkrimi i programeve ndihmëse

3.2.1 Përshkrimi i cd_handling.c dhe cd_handling.h

cd_handling.h - Headeri:

- **Përmbajtja:**
  - Deklaron dy funksione:
    - void cd(char *path); – Funksioni që trajton ndryshimin e direktorisë aktuale në një path të dhënë.
    - void removeSpaces(char *path); – Funksioni që heq hapësirat para dhe pas një stringu të dhënë.
  - Përfshin libraritë e nevojshme: stdio.h, stdlib.h, string.h, unistd.h, dhe ctype.h.

cd_handling.c - Implementimi:

- **Funksioni cd:**
  - **Heqja e hapësirave:** Funksioni fillon duke hequr hapësirat para dhe pas path-it duke thirrur funksionin removeSpaces.
  - **Kontrolli i hapësirave të brendshme:** Nëse path-i përmban hapësira të brendshme, kontrollohet që të jetë e rrethuar nga thonjëza (apo me thonjëza të dyfishta apo të vetme). Nëse nuk është, një mesazh gabimi printohet dhe funksioni kthehet.
  - **Heqja e thonjëzave:** Nëse rruga është e rrethuar nga thonjëza të dyfishta ose të vetme, ato hiqen me strtok().
  - **Kontrolli i rrugës bosh:** Nëse path-i është bosh pas heqjes së hapësirave dhe thonjëzave, funksioni kthehet pa bërë asgjë.
  - **Ndryshimi i drejtorisë:** Funksioni thërret chdir() për të ndryshuar direktorinë në path-in e dhënë. Nëse ka një gabim, një mesazh gabimi printohet.
  - **Shfaqja e direktorisë aktuale:** Pas ndryshimit të direktorisë, funksioni thërret getcwd() për të marrë direktorinë e re dhe e shfaq atë në një mesazh të formatizuar në ngjyrë portkalli.
- **Funksioni removeSpaces:**
  - Ky funksion heq hapësirat që ndodhen para dhe pas një stringu të dhënë (path).
  - **Hapësirat para:** Përdor isspace() për të identifikuar dhe hequr hapësirat para.
  - **Hapësirat pas:** Përdor isspace() për të gjetur dhe hequr hapësirat që ndodhin pas stringut.
  - **Përditësimi i path:** Pas heqjes së hapësirave, funksioni përdor memmove() për të zhvendosur stringun në mënyrë që të mbeten vetëm karakteret e vlefshme.

3.2.2 Përshkrimi i background_handling.c dhe background_handling.h

background_handling.h - Headeri:

- **Përmbajtja:**
  - Deklaron funksionin background, i cili trajton ekzekutimin e komandave në background.

    - Përfshin libraritë e nevojshme si stdio.h, stdlib.h, string.h, unistd.h, dhe sys/wait.h.
- **Funksioni:**
    - void background(char *command); – Deklaron funksionin që merr një varg komandash dhe ekzekuton komandën në background, duke përdorur fork() dhe execvp().

background_handling.c - Implementimi:

- **Përgatitja e komandës:**
    - Komanda ndahet në argumente duke përdorur strtok(), dhe argumentet ruhen në array args.
- **Kontrollimi i &:**
    - Verifikohet nëse komanda përfundon me simbolin &. Nëse po, ai hiqet dhe vendoset një NULL në vendin e tij.
- **Fork dhe exec:**
    - **Në procesin fëmijë (për pid == 0)**, ekzekutohet komanda me execvp pas 5 sekondash, duke kaluar argumentet.Ndërkohë përdoruesi mund të ekzekutojë komanda të tjera.
    - **Në procesin prind**, nuk bëhet asgjë specifike.
- **Gabime:**
    - Nëse ndodh një gabim gjatë fork() (pid < 0), një mesazh gabimi printohet.
    - Nëse komanda nuk ka & në fund, printohet një mesazh gabimi që tregon se komanda nuk është një operacion i vlefshëm në background.

3.2.3 Përshkrimi i execute_handling.c dhe execute_handling.h

execute_handling.h - Headeri:

- **Përmbajtja:**
    - Deklaron funksionin execute, i cili trajton ekzekutimin e komandave në shell.
    - Përfshin libraritë e nevojshme si stdio.h, stdlib.h, string.h, unistd.h, dhe sys/wait.h.
- **Funksioni:**
    - void execute(char *command); – Deklaron funksionin që merr një varg komandash dhe ekzekuton komandën në procesin e ri duke përdorur fork() dhe execvp().

execute_handling.c - Implementimi:

- **Përgatitja e komandës:**
    - Komanda ndahet në argumente duke përdorur strtok() dhe argumentet ruhen në array args.
    - Pas përfundimit të ndarjes, vendoset një NULL në fund të array për të sinjalizuar fundin e listës së argumenteve.
- **Fork dhe exec:**
    - **Në procesin fëmijë (për pid == 0)**, komanda ekzekutohet me execvp(), duke kaluar argumentet në args. Nëse ekzekutimi dështon, printohet një mesazh gabimi dhe procesi fëmijë mbyllet.

    - **Në procesin prind**, waitpid() përdoret për të pritur përfundimin e procesit fëmijë para se të vazhdojë ekzekutimi i kodit të mëtejshëm.
- **Gabime:**
    - Nëse fork() dështon (pid < 0), printohet një mesazh gabimi.
    - Nëse execvp() dështon, printohet një mesazh gabimi dhe procesi fëmijë mbyllet.

3.2.4 Përshkrimi i pipe_handling.c dhe pipe_handling.h

pipe_handling.h - Headeri:

- **Përmbajtja:**
    - Deklaron dy funksione:
        - bool valid_pipe_command(char *command); – Funksioni që kontrollon nëse komanda e dhënë është e vlefshme për përdorimin e pipe.
        - void pipe_process(char *command); – Funksioni që trajton ekzekutimin e komandave që përdorin pipe (|) për të lidhur dy komanda.
    - Përfshin libraritë e nevojshme: stdio.h, stdlib.h, string.h, unistd.h, sys/types.h, sys/wait.h, dhe stdbool.h.

pipe_handling.c - Implementimi:

- **Funksioni valid_pipe_command:**
    - Përdoret për të verifikuar se pipe (|) është i rrethuar nga hapësira në komandën e dhënë.
    - Kjo siguron që komanda të jetë në formatin e duhur për t'u përpunuar me pipe.
- **Funksioni pipe_process:**
    - **Kontrollimi i komandës:** Verifikohet nëse komanda është e vlefshme për përdorimin e pipe, dhe nëse nuk është, printohet një mesazh gabimi.
    - **Përgatitja e komandave:** Komanda ndahet në dy pjesë duke përdorur strtok() për të ndarë komandat e para dhe të dyta, duke i përgatitur për ekzekutim.
    - **Krijimi i tubit:** Krijohet një pipe për të mundësuar kalimin e daljes nga komanda e parë në hyrjen e komandës së dytë.
    - **Fork dhe exec:**
        - **Në procesin e parë (për pid1 == 0)**, komanda e parë ekzekutohet duke përdorur execvp(), duke dërguar output-in e saj në pipe.
        - **Në procesin e dytë (për pid2 == 0)**, komanda e dytë ekzekutohet duke marrë input nga pipe.
    - **Procesi prind:** Pritet përfundimi i të dy proceseve fëmijë përpara se të mbyllet pipe dhe të përfundojë ekzekutimi.
- **Gabime:**
    - Nëse ndodh një gabim gjatë fork() (pid1 < 0 ose pid2 < 0), printohet një mesazh gabimi.
    - Po ashtu, nëse ndodh një gabim gjatë ekzekutimit të komandave, printohet një mesazh gabimi dhe procesi përfundon.

3.2.5 Përshkrimi i redirection_handling.c dhe redirection_handling.h

redirection_handling.h - Headeri:

- **Përmbajtja:**
  - Deklaron dy funksione:
    - void redirection(char *command, char *input_file, char *output_file); – Funksioni që trajton drejtpërdrejtë redirektimin e hyrjes dhe daljes nëpërmjet skedarëve.
    - void redirect_process(char *command); – Funksioni që menaxhon procesin që përfshin redirektimin e hyrjes dhe daljes dhe ekzekuton komandën.
  - Përfshin libraritë e nevojshme: stdio.h, stdlib.h, fcntl.h, unistd.h, string.h, dhe sys/wait.h.

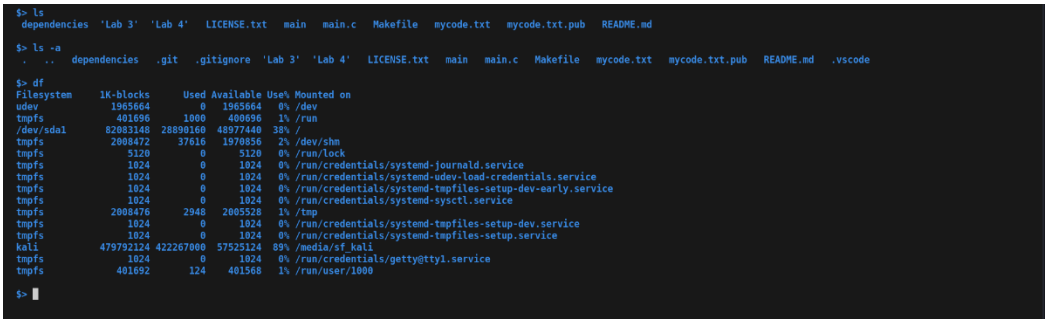
redirection_handling.c - Implementimi:

- **Funksioni redirection:**
  - **Redirektimi i hyrjes (input):** Nëse emri i skedarit të hyrjes (input_file) është i dhënë, hapet skedari për lexim (O_RDONLY). Nëse hapja dështon, një mesazh gabimi printohet dhe funksioni kthehet. Nëse hapja është e suksesshme, dup2() përdoret për të lidhur skedarin e hyrjes me standard input-in (STDIN_FILENO).
  - **Redirektimi i daljes (output):** Po ashtu, nëse emri i skedarit të daljes (output_file) është i dhënë, hapet skedari për shkrim (O_WRONLY | O_CREAT | O_TRUNC). Nëse hapja dështon, një mesazh gabimi printohet dhe funksioni kthehet. Nëse hapja është e suksesshme, dup2() përdoret për të lidhur skedarin e daljes me standard output-in (STDOUT_FILENO).
  - **Mbyllja e deskriptorëve:** Pas lidhjes së deskriptorëve të skedarëve me standardet përkatëse, skedarët mbyllen për të liruar burimet.
- **Funksioni redirect_process:**
  - **Fork:** Krijohet një proces fëmijë për të ekzekutuar komandën.
  - **Përgatitja e argumenteve:** Komanda ndahet në argumente duke përdorur strtok(). Pas kësaj, kërkohet nëse ka operacione redirektimi (< për hyrje dhe > për dalje) dhe, nëse po, ato përpunohen dhe përkatësisht caktohen emrat e skedarëve.
  - **Redirektimi:** Funksioni redirection() thirret për të realizuar redirektimin nëse janë caktuar skedarët për input dhe output.
  - **Ekzekutimi i komandës:** Pas përgatitjes së argumenteve dhe redirektimit, komanda ekzekutohet duke përdorur execvp(). Nëse ekzekutimi dështon, një mesazh gabimi printohet.
  - **Procesi prind:** Pas forkimit, procesi prind pret përfundimin e procesit fëmijë me waitpid().
- **Gabimet:**
  - Nëse ndodh një gabim gjatë forkimit, hapjes së skedarëve, ose ekzekutimit të komandës, printohen mesazhe gabimi dhe procesi ndalet.

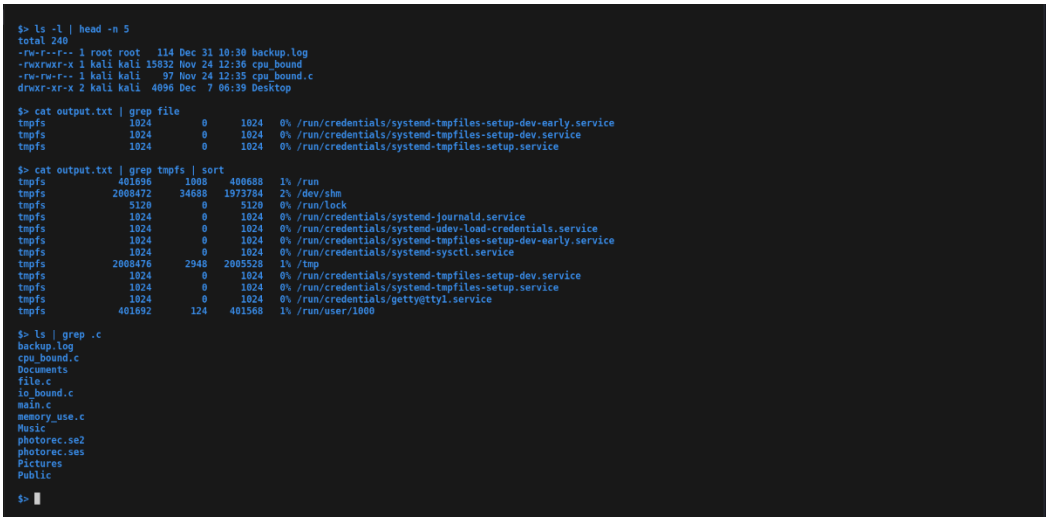
4 Testimi

4.1 Komanda cd

4.2 Komanda të ndryshme

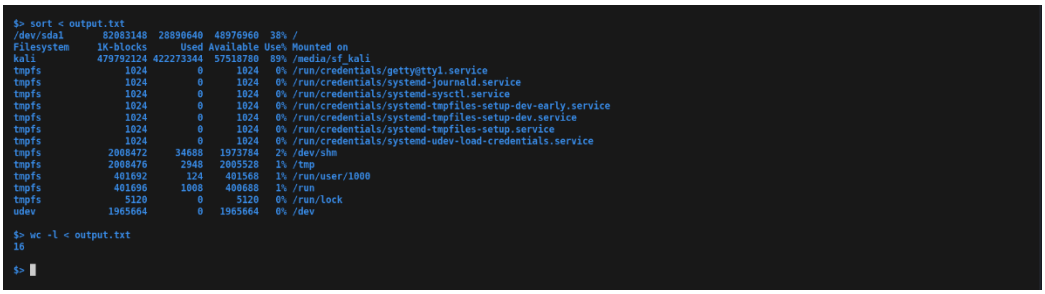


4.3 Komanda me pipe

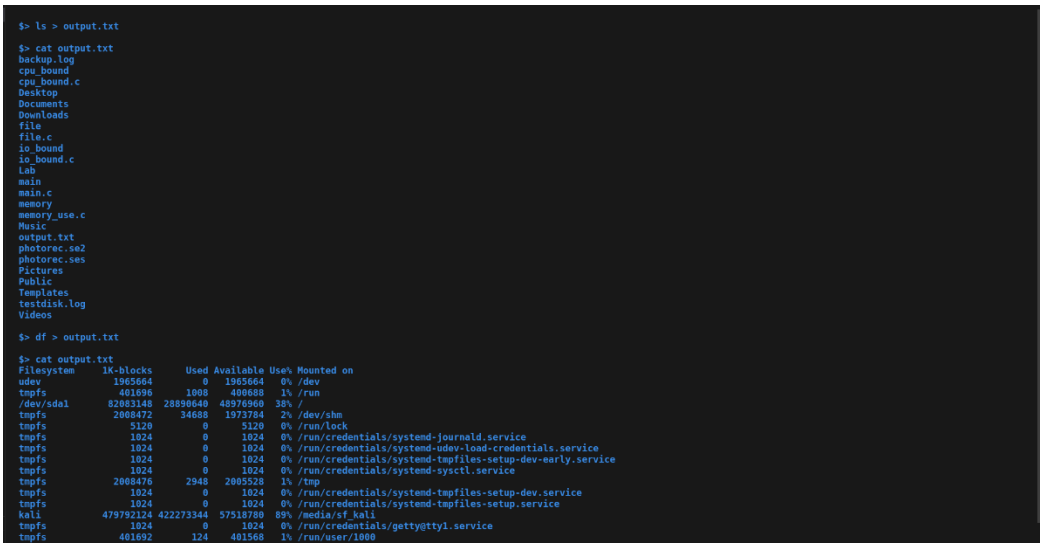


4.4 Komanda me redirection

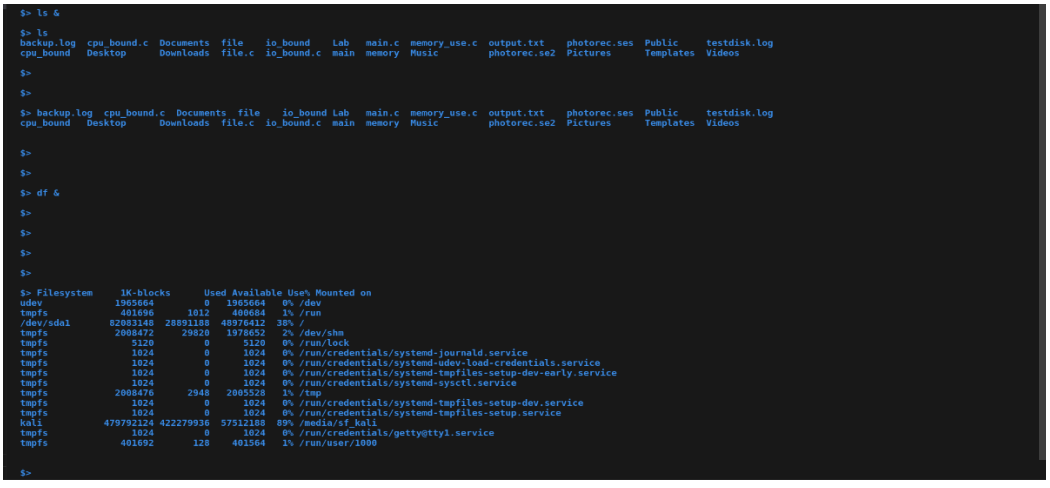
4.4.1 Output



4.4.2 Input



4.5 Komanda në background



5 Konkluzione

Ky program shell i thjeshtë është një mënyrë e lehtë për dalluar dhe demonstruar mënyrën se si përdoruesit mund të veprojnë me sistemin operativ me komanda të thjeshta. Vetë natyra e programin është e lehtë çka e bën shumë të thjeshtë për t'u kuptuar. Programi nuk është i tillë që të mund të zevendësojë programin aktual të shell , por një version shumë më i thjeshtuar i tij.

Shell-i mbulon funksionalitetet bazë duke përfshirë:

- Ekzekutimin e komandave të thjeshta.
- Menaxhimin e komandave në background me simbolin &.
- Përpunimin e komandave me pipe (|) për komunikimin ndërmjet proceseve.
- Redirektimin e input-it dhe output-it me > dhe <.

Struktura e programin është e ndarë ne disa direktori të cilat përmbajnë funksionet e caktuara të ndryshme për komandat. Ndarja e tyre ne direktori të ndryshme është një praktikë e mirë programimi që ndihmon në organizimin dhe mirëmbajtjen e kodit, gjithashtu dhe në ripërdorimin e tij (si në rastin e funkstionit removeSpaces). Duke ndarë funksionalitetet në file dhe direktorë specifikë (p.sh., cd_handling, background_handling, etj.), bëhet më e lehtë të kuptohet struktura e projektit dhe të menaxhohen modulet individuale.

Me ndryshimin e ngjyrave për komandat dhe mesazhet e gabimit, shell-i përmirëson ndërveprimin vizual me përdoruesin duke synuar ta bëjë eksperiencën sa më të ngjashme me shell-in në Unix.

Programi është një demonstrim shumë i mirë i mënyrës se si sistemi operativ komunikon me përdoruesin dhe proceset me njëri-tjetrin.