Dokumentacja Projektu My Shell

SPIS TREŚCI

Opis projektu

Projekt My Shell to prosty emulator powłoki systemowej (shell), który czyta polecenia ze standardowego wejścia (lub z pliku skryptu) i wykonuje je, korzystając z systemowego środowiska uruchomieniowego. Shell wykonuje polecenia przez wywołania systemowe z rodziny exec (np. execl, execlp, execvp), obsługuje przekierowania, potoki oraz inne podstawowe funkcjonalności powłoki.

Funkcjonalności

1. Wykonywanie poleceń: Shell wykonuje polecenia wprowadzone przez użytkownika lub zawarte w skrypcie, rozdzielając je na komendy i argumenty.

2. Przekierowanie wyjścia (>): Możliwość przekierowania standardowego wyjścia polecenia do pliku.

3. Obsługa potoków (|): Tworzenie potoków dowolnej długości, umożliwiające przekazywanie wyjścia jednego polecenia jako wejścia innego.

4. Uruchamianie w tle (&): Polecenia mogą być uruchamiane w tle, co umożliwia kontynuowanie pracy bez czekania na ich zakończenie.

5. Zmiana katalogu roboczego (cd): Funkcja zmiany bieżącego katalogu pracy.

6. Tworzenie plików (touch): Możliwość tworzenia nowych plików.

7. Wyświetlanie historii poleceń: Historia wykonanych poleceń jest zapisywana i może być wyświetlana.

8. Obsługa sygnałów (SIGQUIT): Obsługa sygnałów, np. dla wyświetlenia historii poleceń.

Dokładny Opis Funkcjonalności związanej z tworzeniem plików (touch)

Funkcja touch w powłoce jest zaimplementowana w sposób umożliwiający użytkownikowi tworzenie nowych plików lub aktualizację daty modyfikacji istniejących plików. Wykorzystuje do tego systemowe wywołanie open z flagami O_WRONLY | O_CREAT.

Sygnatura funkcji

Funkcja jest zdefiniowana w następujący sposób:

int check_touch_command(char *args[]);

Parametry i zachowanie

args: Tablica ciągów znaków, gdzie args[0] to touch, a args[1], args[2], ..., args[n] to nazwy plików, które mają być utworzone lub zaktualizowane.

Funkcja sprawdza, czy pierwszy argument to touch. Jeśli tak, przechodzi do przetwarzania dalszych argumentów jako nazw plików.

Dla każdego pliku podanego jako argument, funkcja próbuje otworzyć plik z użyciem flag O_WRONLY (otwarcie pliku do zapisu) i O_CREAT (utworzenie nowego pliku, jeśli nie istnieje).

Wartość mode ustawiona jest na 0644, co oznacza, że plik będzie miał prawa do czytania i pisania przez właściciela oraz prawa do czytania przez grupę i innych.

Obsługa błędów

Funkcja obsługuje błąd EACCES, gdzie plik istnieje, ale nie jest zapisywalny/wykonywalny jak żądano, lub katalog nie pozwala na utworzenie pliku.

EEXIST nie jest bezpośrednio obsługiwany, ponieważ flaga O_CREAT bez O_EXCL nie powoduje błędu, gdy plik już istnieje.

Inne błędy, takie jak EMFILE, ENFILE, ENOENT, ENOSPC, ENXIO, EROFS są obsługiwane przez system i zwracane przez funkcję open z odpowiednimi komunikatami błędów.

Zwracane wartości

Funkcja zwraca 1, jeśli polecenie było poleceniem touch i zostało przetworzone (niezależnie od tego, czy operacja na plikach się powiodła, czy też zwróciła błąd).

Zwraca 0, jeśli pierwszy argument nie jest touch, co pozwala na dalsze przetwarzanie innych poleceń w powłoce.

Przykład użycia w powłoce

touch newfile.txt

Tworzy plik newfile.txt w bieżącym katalogu roboczym lub aktualizuje datę jego ostatniej modyfikacji, jeśli plik już istnieje.

Dokładny Opis Funkcjonalności związanej ze zmianą katalogu roboczego (cd)

Funkcja cd (change directory) w powłoce pozwala użytkownikowi na zmianę bieżącego katalogu roboczego. Funkcja ta jest kluczowym narzędziem w nawigacji po systemie plików w interfejsie wiersza poleceń.

Sygnatura funkcji

Funkcja jest zdefiniowana w następujący sposób:

int check_cd_command(char *args[]);

Parametry i zachowanie

args: Tablica ciągów znaków, gdzie args[0] to cd, a args[1] to ścieżka katalogu, do którego użytkownik chce się przełączyć.

Funkcja sprawdza, czy pierwszy argument to cd. Jeśli tak, to przetwarza drugi argument jako ścieżkę docelową.

Jeśli args[1] jest NULL, co oznacza, że brakuje argumentu ścieżki, funkcja wypisuje komunikat o błędzie: "cd: argument expected".

Obsługa błędów

Jeśli funkcja chdir (zmiana katalogu) nie powiedzie się, wywołuje perror z komunikatem "cd", który informuje użytkownika o rodzaju napotkanego problemu. Możliwe błędy to:

EACCES: Brak dostępu do katalogu.

ENOENT: Katalog nie istnieje.

ENOTDIR: Wskazana ścieżka nie jest katalogiem.

ELOOP: Zbyt wiele poziomów dowiązań symbolicznych przy próbie dostępu do katalogu.

Zwracane wartości

Funkcja zwraca 1 jeśli polecenie było poleceniem cd i zostało przetworzone, co pozwala przerwać dalsze przetwarzanie innych potencjalnych poleceń.

Zwraca 0 jeśli pierwszy argument nie jest cd, umożliwiając systemowi przetwarzanie innych poleceń.

Przykład użycia w powłoce

cd /path/to/directory

Zmienia bieżący katalog roboczy na /path/to/directory. Jeśli taki katalog nie istnieje lub użytkownik nie ma do niego dostępu, wyświetlany jest odpowiedni komunikat błędu.

Uruchamianie projektu

Projekt można skompilować i uruchomić przy użyciu standardowych narzędzi w środowisku Unix/Linux. Poniżej znajduje się przykład komend do kompilacji i uruchomienia:

make

następnie

./my_shell

Testowanie projektu

Skrypty do testowania

Skrypt nr 1: Testowanie podstawowych poleceń (test_basic_commands.sh)

#!/home/sciezka/my_shell

ls

ls -a

echo "Wyświetlam aktualny katalog:"

pwd

```
echo "Lista użytkowników:"

who
```

Skrypt nr 2: Testowanie działania funkcji cd (test_cd_command.sh)

```
#!/home/sciezka/my_shell

echo "Aktualny katalog:"

pwd

cd ..

echo "Katalog po zmianie:"

pwd

cd /tmp

echo "Katalog po zmianie na /tmp:"

pwd
```

Skrypt nr 3: Działanie skryptu w tle (test_background.sh)

```
#!/home/sciezka/my_shell

echo "Uruchamianie sleep na 10 sekund w tle"

sleep 10 &

echo "Uruchamiam ls:"

ls
```

Skrypt nr 4: Przekierowanie wyjścia (test_redirection.sh)

```
#!/home/sciezka/my_shell

echo "Zapisuje listę plików do pliku files.txt"

ls > files.txt
```

```
echo "Zawartość pliku files.txt:"

cat files.txt
```

Skrypt nr 5: Tworzenie potoków dowolnej długości (test_pipes.sh)

```
#!/home/sciezka/my_shell

echo "Połączenie kilku poleceń w potok:"

ls -l | grep ".txt" | sort
```

Skrypt nr 6: Polecenie touch, nano, cat (test_file_operations.sh)

```
#!/home/sciezka/my_shell

echo "Tworzę nowy plik tekstowy"

touch testfile.txt

echo "Edytuję plik w nano (zamknij, aby kontynuować)"

nano testfile.txt

echo "Wyświetlam zawartość pliku testfile.txt"

cat testfile.txt
```

Skrypt nr 7: Wyświetlenie historii poleceń (test_history.sh)

```
#!/home/sciezka/my_shell

echo "Wysyłam sygnał SIGQUIT do shella, aby wyświetlić historię"

kill -SIGQUIT $$
```

Sposób uruchamiania skryptów

Aby uruchomić którykolwiek ze skryptów, trzeba nadać mu prawa do wykonania poleceniem
chmod +x nazwa_skryptu.sh
a następnie uruchomić poleceniem:
./nazwa_skryptu.sh