Documentação do Projeto: Shell Simplificado em C

Luis Henrique Gomes Higino Luisa Lopes Carvalhaes

October 2, 2025

1 Soluções Implementadas

1.1 Task 1: Criação de Processos

A função `fork1` é responsável por criar novos processos utilizando a syscall `fork`. Caso a criação falhe, a função imprime uma mensagem de erro utilizando `perror` e encerra o programa. Esta abordagem garante robustez, pois evita que erros de fork passem despercebidos.

```
int fork1(void) {
    pid_t pid = fork();
    if (pid < 0) {
        perror("fork");
        exit(1);
    }
    return pid;
}
```

Listing 1: Função fork1

1.2 Task 2: Execução de Comandos Simples

A função `handle_simple_cmd` executa comandos simples, substituindo o processo atual pelo processo do comando via `execvp`. O uso de `execvp` foi escolhido por sua compatibilidade com o armazenamento de argumentos no struct `execcmd`.

```
void handle_simple_cmd(struct execcmd *ecmd) {
    char *command_name = ecmd->argv[0];
    char **args = ecmd->argv;
    execvp(command_name, args);
    perror("execvp");
    exit(1);
}
```

Listing 2: Função handle_simple_cmd

1.3 Task 3: Redirecionamento de Entrada/Saída

A função `handle_redirection` abre o arquivo especificado e redireciona o descritor de arquivo usando `dup2`. Após o redirecionamento, o descritor original é fechado.

```
void handle_redirection(struct redircmd *rcmd) {
    int fd = open(rcmd->file, rcmd->mode, 0644);
    if (fd < 0) { perror("open"); exit(1); }
    if (dup2(fd, rcmd->fd) < 0) { perror("dup2"); close(fd); exit(1); }
    close(fd);
```

```
}
```

Listing 3: Função handle_redirection

1.4 Task 4: Criação e Manipulação de Pipes

A função `handle_pipe` cria um pipe entre dois processos filhos. O filho esquerdo escreve no pipe, enquanto o direito lê dele. O pai fecha os descritores e aguarda os filhos terminarem.

```
void handle_pipe(struct pipecmd *pcmd, int *p, int r) {
    int pf[2];
    if (pipe(pf) < 0) { perror("pipe"); exit(1); }
    int pid1, pid2;

    if ((pid1 = fork1()) == 0) {
      dup2(pf[1], STDOUT_FILENO);
      close(pf[0]); close(pf[1]);
      runcmd(pcmd->left);
    } else if ((pid2 = fork1()) == 0) {
      dup2(pf[0], STDIN_FILENO);
      close(pf[0]); close(pf[1]);
      runcmd(pcmd->right);
    }

    close(pf[0]); close(pf[1]);
    wait(&pid1); wait(&pid2);
}
```

Listing 4: Função handle_pipe

1.5 Task 5: Comando Interno `cd`

O comando `cd` precisa ser executado no processo atual, pois alterar o diretório em um filho não muda o diretório do pai. A função `chdir` retorna erro caso o diretório não exista ou não seja permitido o acesso, e a mensagem de erro foi ajustada para refletir isso.

2 Estruturas de Dados de Comandos

- `struct cmd`: tipo genérico de comando.
- `struct execcmd`: comando de execução simples.
- `struct redircmd`: comando com redirecionamento de entrada/saída.
- `struct pipecmd`: comando com pipe.

3 Referências Bibliográficas

- Slides das aulas do curso
- Páginas de manual do Linux