Documentação do Trabalho 1

Davi Carvalho dos Santos - 2022035580
Davi Sakamoto Lamounier - 2022035873

Descrição do Sistema

O sistema desenvolvido consiste em dois componentes principais: um shell simplificado (parte1) e uma ferramenta de monitoramento de processos (parte2). O shell permite a execução de comandos do sistema, incluindo suporte a redirecionamento de arquivos e pipes. A ferramenta de monitoramento exibe informações sobre processos em execução no sistema e permite o envio de sinais a esses processos. Ambos os componentes foram implementados utilizando a linguagem C.

Módulo 1: Shell Simples (parte1)

Objetivo

O shell implementado permite ao usuário executar comandos no sistema, com suporte a:

- Execução de comandos simples.
- Redirecionamento de entrada e saída.
- Encadeamento de comandos com pipes.

Funcionalidade

O shell interpreta os comandos fornecidos pelo usuário e executa-os conforme o tipo de comando. Os tipos de comando suportados são:

- **Execução de Comando Simples**: O comando é executado diretamente com os argumentos fornecidos.
- **Redirecionamento de Arquivo**: O shell permite redirecionar a entrada (<) ou a saída (>) para arquivos.
- **Pipes**: Permite encadear comandos através de pipes (|), passando a saída de um comando como entrada para outro.

Estrutura de Dados

As estruturas de dados principais são:

- cmd: estrutura base para comandos, contendo o tipo do comando.
- execcmd: estrutura para comandos simples de execução.
- redircmd: estrutura para comandos com redirecionamento.
- pipecmd: estrutura para comandos com pipes.

Funções Principais

- **runcmd()**: Executa o comando dependendo do seu tipo (simples, com redirecionamento ou com pipe).
- **parsecmd()**: Analisa e processa a linha de comando, gerando a estrutura correspondente.
- **getcmd()**: Obtém o comando digitado pelo usuário.
- **gettoken()**: Extrai tokens da linha de comando.

Fluxo de Execução

1. O usuário digita um comando.
2. O comando é analisado e convertido em uma estrutura cmd apropriada.
3. O comando é executado ou, se necessário, redirecionado ou encadeado com outro comando.
4. O shell aguarda a conclusão do comando e solicita uma nova entrada.

Módulo 2: Monitoramento de Processos (parte2)

Objetivo

Este módulo oferece uma visualização semelhante ao comando top, exibindo informações sobre os processos em execução no sistema. O usuário pode enviar sinais para os processos utilizando um formato específico de entrada.

Funcionalidade

A ferramenta exibe uma tabela com as seguintes informações para cada processo:

- **PID**: Identificador do processo.
- **User**: Usuário proprietário do processo.
- **PROCNAME**: Nome do processo.
- **Estado**: Estado atual do processo (como R para rodando).

Além disso, o usuário pode enviar sinais a um processo específico digitando seu PID e o número do sinal.

Estrutura de Dados

- **ProcessInfo**: Estrutura que contém informações sobre um processo, como PID, nome do usuário, nome do processo e estado.
- **processes**: Array de ProcessInfo para armazenar até 20 processos.

Funções Principais

- **get_username()**: Obtém o nome do usuário a partir do UID.

- **fetch_process_info()**: Coleta informações sobre os processos em execução a partir do diretório /proc.
- **display_processes()**: Exibe as informações dos processos em formato de tabela.
- **update_display()**: Thread responsável por atualizar a tabela de processos a cada 1 segundo.
- **handle_input()**: Thread que captura a entrada do usuário para envio de sinais.

Fluxo de Execução

1. O monitor coleta informações dos processos e exibe-as periodicamente.
2. O usuário pode digitar um PID e um sinal para enviar a um processo.
3. O programa continua a exibir as informações dos processos até que o usuário escolha sair.

Compilação e Execução

Compilação

Para compilar os módulos, utilize o seguinte comando:

gcc -o shell parte1/sh.c
gcc -o meutop parte2/meutop.c -lpthread

Execução

1. Para executar o shell simples:

./shell

2. Para executar o monitor de processos:

./meutop

Dependências

- O módulo de monitoramento de processos requer permissões para acessar o diretório /proc e informações de processo.
- O monitoramento e o envio de sinais são suportados no sistema Unix/Linux.

Considerações Finais

Este trabalho implementa dois sistemas distintos, mas complementares, que demonstram o uso de manipulação de processos, redirecionamento e sinalização no sistema operacional. A implementação foi realizada utilizando a linguagem C, com foco no uso de threads, manipulação de arquivos e processos do sistema.