Projeto de Sistemas Operacionais

Shell básico: **shellvis**

O objetivo deste trabalho é desenvolver um interpretador de comandos (Shell) em linguagem C.

Conceitos envolvidos

- Chamadas de sistema
- Criação e gerenciamento de processos
- Gerenciamento de arquivos

Funcionalidade Básica

O programa deve executar em laço contínuo, recebendo comandos de duas formas:

1. Interativamente, via terminal.
2. A partir de um arquivo em modo batch.

Cada comando (cada linha) deve ser interpretado como:

1. Comando interno (*built-in*).
2. Programa externo.

A execução ocorre em um processo filho, repetindo-se até que o usuário digite `exit`.

Modos de Execução

Existem dois modos de operação:

- **Interativo**: o usuário digita comandos no terminal. O prompt exibido deve ser:

  ```
  shellvis>
  ```

- **Batch**: os comandos são lidos de um arquivo de entrada:

  ```
  prompt> ./shellvis batch.txt
  ```

  Nesse modo, **nenhum prompt** é exibido. O tratamento deve ser feito na função `mostra_terminal()`.

Os argumentos dos comandos são separados por espaços. Não é necessário lidar com caracteres de escape ou argumentos entre aspas.

Comandos Internos

Comandos internos não devem ser tratados como programas externos. O `shellvis` deve implementar:

- `exit`: encerra o shell.
- `cd <dir>`: altera o diretório atual para `<dir>`
- `path <caminho> [<caminho> ...]`: define os diretório(s) de busca de executáveis
- `pwd`: exibe o caminho absoluto do diretório atual
- `ls`: lista o conteúdo do diretório atual, suportando as opções `-l` e `-a` conforme o funcionamento do ls original
- `cat <arquivo>`: imprime o conteúdo de `<arquivo>`. O binário `cat <arquiv>` lê o conteúdo do arquivo no argumento e o escreve na saída padrão.

Redirecionamento

O `shellvis` deve suportar redirecionamento de entrada e saída.

- Saída para arquivo (comandos internos e programas externos):

```
ls > output.out
```

- Entrada de arquivo (para programas externos):

```
./prog < input.in
```

Programas Externos

Execução de programas externos deve seguir a forma:

```
shellvis> ./prog
```

Programas externos também podem ser executados com argumentos da seguinte maneira:

```
shellvis> ./prog arg1 arg2
```

- Para executar o shell deverá procurar, em todos os caminhos definidos com o `built-in path`, por um executável com o nome inserido no comando
- O programa deve ser executado em processo filho, recebendo os argumentos passados (se tiver)
- Ao término, o shell deve exibir o valor de retorno do programa (se tiver)

O redirecionamento também deve ser aplicado a programas externos.

Comandos em Paralelo

O `shellvis` deve permitir execução concorrente de múltiplos comandos com o operador `&`.

Exemplo:

```
shellvis> ./prog1 & ./prog2 arg1 & ./prog3 < input.in
```

Cada comando deve ser executado em processo separado, de forma paralela.

Resumo da Implementação

Comando	Funcionalidade
`ls`	lista os arquivos e diretórios do diretório atual
`cd <dir>`	altera o diretório atual para `<dir>`
`pwd`	mostra o caminho absoluto do diretório atual
`exit`	encerra o shell
`./prog`	executa o programa `prog`
`./prog arg1 arg2`	executa `prog` com argumentos
`./prog1 > output`	redireciona saída de `prog1` para `output`
`./prog1 < input`	redireciona entrada de `prog1` a partir de `input`

Tratamento de Erros

O `shellvis` deve tratar erros de forma consistente, exibindo mensagens claras e informativas.

Entrega

O material entregue deve incluir:

1. Código-fonte (sem executáveis)
2. Instruções de compilação
3. Conjunto de testes, com entradas e saídas esperadas
4. Relatório breve contendo:
    - Visão geral do código e fluxo de execução do programa
    - Lista de comandos implementados (ou não, nesse caso adicionar justificativo do por que não conseguiu implementar)
    - Descrição da abordagem de implementação de cada comando

Todos os arquivos devem ser reunidos em um único `.zip` e enviados pelo Classroom até a data limite.