

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 (até que o usuário digite **exit**), 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 de programas externos deve ocorrer em um processo filho. Built-ins podem executar no mesmo processo pai.

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
```

Em ambos os modos, os argumentos dos comandos são separados por espaço(s). Não é necessário lidar com caracteres de escape ou argumentos entre aspas.

Comandos Internos (Built-in)

Comandos internos são implementados como parte do código do `shellvis` e podem ser executados no mesmo processo pai. Você deve implementar os seguintes comandos:

- `exit`: encerra o `shellvis`
- `cd <dir>`: altera o diretório atual para `<dir>`
- `path <caminho> [<caminho> ...]`: define a lista de diretórios onde o shell procura executáveis
- `pwd`: imprime o caminho absoluto do diretório atual

Programas Externos

Programas externos são executáveis que residem no sistema de arquivos. O `shellvis` deve ser capaz de executar dois tipos de programas externos:

1. Binários implementados como parte do projeto
2. Programas externos já existentes no sistema (ex: `/bin/echo`)

A execução deve seguir o seguinte formato:

```
shellvis> nome_do_programa [arg1] [arg2] ...
```

O `shellvis` deve procurar pelo `nome_do_programa` em todos os diretórios definidos com o `built-in path`.

A execução deve ocorrer em um processo filho, que recebe os argumentos passados na linha de comando.

Como parte deste projeto, você deve criar os seguintes programas executáveis. É necessários que esses programas sejam implementados utilizando chamadas de sistema diretamente para a manipulação de arquivos, e não apenas encapsulando (chamando por trás) os comandos originais do sistema.

Os programas externos para serem implementados são:

- `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
- `grep [-n] <string> <arquivo>` (grep simplificado): procura pela `<string>` dentro do `<arquivo>` e escreve na saída padrão todas as linhas em que a string aparece. Não é necessário implementar suporte a expressões regulares (regex)
- `touch <arquivo>`: cria um arquivo vazio se ele não existir, ou atualiza a data de modificação se ele já existir
- `rm <arquivo>`: apaga um arquivo
- `cp <origem> <destino>`: copia o conteúdo do arquivo de `<origem>` para um arquivo `<destino>`

Redirecionamento de Saída

O `shellvis` deve suportar o redirecionamento de entrada (`<`) e saída (`>`)

- **Saída para arquivo**

```
shellvis> ls -l > saida.txt
```

- **Entrada de arquivo**

```
shellvis> ./prog < log.txt
```

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 separado por **&** deve ser executado em um processo filho distinto, de forma paralela.

Tratamento de Erros

O **shellvis** deve tratar erros de forma consistente, exibindo mensagens claras e informativas para o usuário. Exemplos de erros a serem tratados:

- Comando não encontrado
- Arquivo de entrada para redirecionamento inexistente
- Diretório inválido para o comando **cd**

Entrega

O material entregue deve incluir:

1. Código-fonte (sem executáveis do shellvis)
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.