Bruno Fernando da Silva

Lucas de Jesus Lima

Relatório do Projeto Terminal

Classes e funcionalidades

Diretório: Classe que representa o objeto diretório no simulador. É composta dos seguintes atributos:

private nome (String): Indica o nome do diretório.

private local (String): Indica o local do diretório.

private path_unico(String): Indica o caminho do diretório.

private datacriacao (long): Atributo responsável por armazenar a data da criação do diretório.

private pai (Diretorio): Armazena objeto de diretório que é o pai do atual diretório.

private arquivos(ArrayList<Arquivo>): ArrayList do tipo Arquivo que armazena coleções de arquivos que estão introduzidos a um diretório.

private diretorios(ArrayList<Diretorio>): ArrayList do tipo Diretorio que armazena coleções de diretórios que estão afilados a um diretório.

private proprietario(String): String que armazena o nome do proprietário do diretório atual.

private permissao(String):String responsável por armazenar a especificação sobre a permissão do diretório.

Observação: A especificação de local path foi criada como uma solução para problemas em relação a busca de pastas onde tenha uma pasta com o mesmo nome antes da pasta desejada.

Arquivo: Classe que representa o objeto arquivo no simulador. É composta dos seguintes atributos:

path (Diretorio): Representa o caminho no qual o arquivo se encontra.

private nome (String): String que representa o nome do arquivo.

private proprietario (String): String que representa o nome do proprietario do arquivo.

private permissao (String): String responsável por armazenar a especificação sobre a permissão do arquivo atual.

private datacriacao (Date): Date que armazena a data com hora e segundos da criação do arquivo.

private datamodificacao (Date): Date que armazena a data com hora e segundos do momento que foi feita a última modificação no arquivo, sendo a primeira o momento de sua criação.

private ultimoacesso (Date): Date que armazena a data com hora e segundos do momento que o arquivo foi acessado pela ultima vez.

conteudo (String): String responsável por armazena o conteúdo de texto do arquivo.

Permissões: Classe responsável por incorporar as permissões do usuário. É composta pelos seguintes atributos:

private usuario (String): String responsável por armazenar o nome do usuário

private permissao (int): Inteiro responsável por armazenar o valor referente a permissão.

Usuario: Classe que representa o objeto do usuário no sistema. É composta pelos seguintes atributos:

private nome (String): String responsável por armazenar o nome do usuário.

private senha (String): String responsável por armazena a senha do usuário.

SO: Classe responsável por inicializar e estabelecer os fundamentos base do simulador, uma vez que essa estrutura de dados será a do Linux devido a facilidade e familiarização com a estrutura assim como o controle de acesso de arquivos. Possui os métodos:

recebe_comandos: Recebe da main Terminal a entrada do usuário e encaminha para o comando indicado na classe Comandos.

inicializar: É responsável por criar o usuário root e a estrutura base de diretórios.

SO: Inicia a estrutura de diretórios padrão e chama a função inicializar.

zera_raiz: zera a raiz dos diretórios.

Além dessas, possui função padrões para enviar e receber dados de atributos (getters e setters).

É composta pelos seguintes atributos:

private hostname (String): String estática responsável por armazenar o nome do computador.

private usuário_atual (String): String estática que é responsável por armazenar o nome do usuário atual que inicialmente é o root.

Atributos relacionados a estrutura básica de arquivo e diretório:

private users (ArrayList<Usuario>): ArrayList estática do tipo Usuario, responsável por armazenar uma coleção de usuários.

private pasta (ArrayList<Diretorio>): ArrayList estática do tipo Diretorio responsável por armazenar uma coleção de diretórios.

private arquivo (ArrayList<Arquivo>): ArrayList estática do tipo Arquivo responsável por armazenar uma coleção de arquivos.

Atributos essenciais para a navegação:

private path_atual (String): Caminho atual do diretório.

private Dir_Atual (Diretorio): Atributo estático que irá armazenar o diretório atual.

private Dir_Inicial (Diretorio): Atributo estático que irá armazenar o diretório inicial.

Comandos: Classe que carrega todos os métodos dos comandos do terminal, é carregada e chamada pela classe SO. Possui vários métodos, sendo alguns auxiliares para os comandos como o navega que encontra um endereço desejado para fazer as operações desejadas.

Lista dos comandos:

tree: Imprime todos os diretórios existentes no diretório atual e em seu subdiretórios.

exit: Comando que finaliza o simulador e a execução do programa.

mkdir: Comando que cria diretórios, quando acompanhado apenas de uma palavra é criado o diretório com o esse nome, caso seja indicando o caminho com os subdiretórios existentes, é criado o novo diretório nesse local.

adduser: Comando que cria um novo usuário, recebe por parâmetro o nome e verifica se já existe usuário com esse nome, caso não existir é solicitado a senha e a confirmação dela para assim criar o novo usuário e alocando um diretório para ele na home.

passwd: Comando que altera a senha do usuário, caso existir um usuário com aquele nome é pedido a nova senha e a confirmação de senha.

crfile: Comando que criar um novo arquivo no diretório atual. Por padrão é criado apenas arquivo de texto, é especificado o nome do arquivo e em seguida o conteúdo dele.

cd: Comando que alterna entre os diretórios atuais, caso a palavras que seguem cd for de um subdiretório do diretório atual, ele se tornará o novo diretório atual, também pode percorrer com barras especificando mais subdiretórios. Caso for acompanhado de “ ..” o diretório atual se torna o diretório pai.

ls: Comando que lista todos os subdiretórios do diretório atual.

removeuser: Acompanhada do nome é feito a busca de um usuário com esse nome caso encontrado ele será removido assim como seu diretório na pasta home.

ren: Renomeia um diretório ou um arquivo especificado por outro nome também especificado, caso existir um arquivo ou diretório com o mesmo nome do que será alterado a ação é impedida e uma mensagem é mostrada na tela indicando a duplicidade.

rm: Remove um arquivo ou diretório especificado, isso se ele existir no diretório atual.

append: Se o nome do arquivo especificado existir no diretório atual, um nova linha é lida sendo essa acoplada ao conteúdo do arquivo existente

chmod: Recebe especificado a permissão sendo 0 o usuário não pode ler nem escrever, 1 somente leitura, 2 leitura e escrita, passa a permissão para determinado usuário em relação ao caminho também especificado.

hostname: Altera o hostname do sistema.

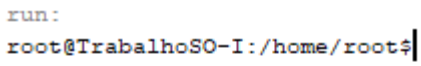
properties: Recebe o nome do arquivo ou diretório e lista então suas permissões se tiverem sido atribuídas a ele.

cat: Recebe o nome de um arquivo, se esse arquivo estiver no diretório atual é exibido na tela todo seu conteúdo.

format: Realiza uma formatação lógica no sistema de arquivos, utilizando as funções zera_raiz e inicia_raiz.

Demonstração de telas:

Tela Inicial



Por padrão mostrando o usuário root logado, o hostname TrabalhoSO-I e o caminho do diretório atual.

Comando adduser:

```
run:
root@TrabalhoSO-I:/home/root$adduser teste
Informe a senha: 123
Confirme a senha: 123
root@TrabalhoSO-I:/home/root$
```

A palavra que segue o comando se torna o nome do usuário, é exigido uma senha e a confirmação para que o usuário seja criado com sucesso. É criado na pasta home um diretório com o nome do usuário.

Comando removeuser:

```
run:
root@TrabalhoSO-I:/home/root$adduser teste
Informe a senha: 123
Confirme a senha: 123
root@TrabalhoSO-I:/home/root$removeuser teste
root@TrabalhoSO-I:/home/root$
```

O usuário que foi acabado de criar é excluído assim como sua pasta na pasta home.

Comando cd:

```
run:
root@TrabalhoSO-I:/home/root$adduser teste
Informe a senha: 123
Confirme a senha: 123
root@TrabalhoSO-I:/home/root$removeuser teste
root@TrabalhoSO-I:/home/root$ls
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$
```

Comando cd pode ser utilizado de 3 formas diferentes, a primeira sendo o da imagem acima onde inserindo “ ..” o diretório atual se torna o diretório pai.

```
run:
root@TrabalhoSO-I:/home/root$adduser teste
Informe a senha: 123
Confirme a senha: 123
root@TrabalhoSO-I:/home/root$removeuser teste
root@TrabalhoSO-I:/home/root$ls
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd root
root@TrabalhoSO-I:/home/root$
```

A segunda forma é utilizando o nome do diretório que se deseja encaminhar, uma vez que esse diretório faça parte dos subdiretórios do diretório atual.

```
run:
root@TrabalhoSO-I:/home/root$adduser teste
Informe a senha: 123
Confirme a senha: 123
root@TrabalhoSO-I:/home/root$removeuser teste
root@TrabalhoSO-I:/home/root$ls
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd root
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd ..
root@TrabalhoSO-I:/$cd home/root
root@TrabalhoSO-I:/home/root$
```

Nessa terceira forma, foi voltado 2 diretórios para pode ser executado. É inserido o caminho direto que deseja se redirecionar, podendo assim entrar em um subdiretório de um diretório de uma forma mais prática.

Comando crfile:

```
run:
root@TrabalhoSO-I:/home/root$adduser teste
Informe a senha: 123
Confirme a senha: 123
root@TrabalhoSO-I:/home/root$removeuser teste
root@TrabalhoSO-I:/home/root$ls
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd root
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd ..
root@TrabalhoSO-I:/$cd home/root
root@TrabalhoSO-I:/home/root$crfile arquivo
Informe o Conteudo: conteudo de um arquivo
root@TrabalhoSO-I:/home/root$
```

É criado um arquivo de texto com o nome arquivo, e na linha abaixo é exigido seu conteúdo, o arquivo é inserido no diretório atual.

Comando ls:

```
Informe a senha: 123
Confirme a senha: 123
root@TrabalhoSO-I:/home/root$removeuser teste
root@TrabalhoSO-I:/home/root$ls
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd root
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd ..
root@TrabalhoSO-I:/$cd home/root
root@TrabalhoSO-I:/home/root$crfile arquivo
Informe o Conteudo: conteudo de um arquivo
root@TrabalhoSO-I:/home/root$ls
        arquivo
root@TrabalhoSO-I:/home/root$
```

É listado todos arquivos e diretórios existentes no diretório atual, no caso o arquivo recentemente criado.

Comando mkdir:

```
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd root
root@TrabalhoSO-I:/home/root$cd ..
root@TrabalhoSO-I:/home$cd ..
root@TrabalhoSO-I:/$cd home/root
root@TrabalhoSO-I:/home/root$crfile arquivo
Informe o Conteudo: conteudo de um arquivo
root@TrabalhoSO-I:/home/root$ls
        arquivo
root@TrabalhoSO-I:/home/root$mkdir pasta
root@TrabalhoSO-I:/home/root$ls
dir     pasta
        arquivo
root@TrabalhoSO-I:/home/root$
```

Utilizando o comando foi criada um novo diretório dentro do diretório atual, o comando ls foi utilizado para realizar a listagem.

Comando ren:

```
root@TrabalhoSO-I:/$cd home/root
root@TrabalhoSO-I:/home/root$crfile arquivo
Informe o Conteudo: conteudo de um arquivo
root@TrabalhoSO-I:/home/root$ls
        arquivo
root@TrabalhoSO-I:/home/root$mkdir pasta
root@TrabalhoSO-I:/home/root$ls
dir     pasta
        arquivo
root@TrabalhoSO-I:/home/root$ren pasta novapasta
root@TrabalhoSO-I:/home/root$ls
dir     novapasta
        arquivo
root@TrabalhoSO-I:/home/root$
```

O comando renomeia a pasta que foi acabada de criar, primeiro especificando a pasta e depois o seu novo nome. Também pode ser utilizado com arquivos.

Comando rm:

```
root@TrabalhoSO-I:/home/root$ls
        arquivo
root@TrabalhoSO-I:/home/root$mkdir pasta
root@TrabalhoSO-I:/home/root$ls
dir     pasta
        arquivo
root@TrabalhoSO-I:/home/root$ren pasta novapasta
root@TrabalhoSO-I:/home/root$ls
dir     novapasta
        arquivo
root@TrabalhoSO-I:/home/root$rm novapasta
root@TrabalhoSO-I:/home/root$ls
        arquivo
root@TrabalhoSO-I:/home/root$
```

O comando removeu o diretório novapasta, o ls foi usado para mostrar a alteração nos diretórios locais. Pode ser utilizado também com arquivos.

Comando append:

```
root@TrabalhoSO-I:/home/root$mkdir pasta
root@TrabalhoSO-I:/home/root$ls
dir     pasta
        arquivo
root@TrabalhoSO-I:/home/root$ren pasta novapasta
root@TrabalhoSO-I:/home/root$ls
dir     novapasta
        arquivo
root@TrabalhoSO-I:/home/root$rm novapasta
root@TrabalhoSO-I:/home/root$ls
        arquivo
root@TrabalhoSO-I:/home/root$append arquivo
continuação do arquivo
root@TrabalhoSO-I:/home/root$
```

O comando inseriu no arquivo de texto arquivo um novo conteúdo, acoplando com o conteúdo já existente no arquivo.

Comando chmod:

```
dir     novapasta
        arquivo
root@TrabalhoSO-I:/home/root$rm novapasta
root@TrabalhoSO-I:/home/root$ls
        arquivo
root@TrabalhoSO-I:/home/root$append arquivo
continuação do arquivo
root@TrabalhoSO-I:/home/root$chmod 1 root home/root/arquivo
root@TrabalhoSO-I:/home/root$properties arquivo
root@TrabalhoSO-I:/home/root$chmod 1 root arquivo
root@TrabalhoSO-I:/home/root$properties arquivo
        root : 1
root@TrabalhoSO-I:/home/root$chmod 1 root arquivo
root@TrabalhoSO-I:/home/root$
```

Comando utilizado para alterar permissão de um usuário para o especificado arquivo, também podendo ser um diretório.

Comando properties:

```
root@TrabalhoSO-I:/home/root$rm arquivo
root@TrabalhoSO-I:/home/root$crfile arquivo
Informe o Conteudo: conteudo de arquivo
root@TrabalhoSO-I:/home/root$append arquivo
continuacao do conteudo
root@TrabalhoSO-I:/home/root$chmod 1 root arquivo
root@TrabalhoSO-I:/home/root$properties arquivo
        root : 1
root@TrabalhoSO-I:/home/root$
```

É exibido na tela as permissões associadas ao arquivo, também pode ser usado a diretórios.

Comando cat:

```
        root : 1
root@TrabalhoSO-I:/home/root$rm arquivo
root@TrabalhoSO-I:/home/root$crfile arquivo
Informe o Conteudo: conteudo de arquivo
root@TrabalhoSO-I:/home/root$append arquivo
continuacao do conteudo
root@TrabalhoSO-I:/home/root$chmod 1 root arquivo
root@TrabalhoSO-I:/home/root$properties arquivo
        root : 1
root@TrabalhoSO-I:/home/root$cat arquivo

conteudo de arquivo continuacao do conteudo

root@TrabalhoSO-I:/home/root$
```

É exibido todo conteúdo do arquivo de texto arquivo e mostrado na tela.

Comando format:

```
root@TrabalhoSO-I:/home/root$crfile arquivo
Informe o Conteudo: conteudo de arquivo
root@TrabalhoSO-I:/home/root$append arquivo
continuacao do conteudo
root@TrabalhoSO-I:/home/root$chmod 1 root arquivo
root@TrabalhoSO-I:/home/root$properties arquivo
        root : 1
root@TrabalhoSO-I:/home/root$cat arquivo

conteudo de arquivo continuacao do conteudo

root@TrabalhoSO-I:/home/root$format
root@TrabalhoSO-I:/home/root$ls
root@TrabalhoSO-I:/home/root$
```

É formatado o sistema de arquivos, o comando ls foi utilizado para mostrar a ausência do arquivo criado

.

Comando exit:

```
root@TrabalhoSO-I:/home/root$append arquivo
continuacao do conteudo
root@TrabalhoSO-I:/home/root$chmod 1 root arquivo
root@TrabalhoSO-I:/home/root$properties arquivo
        root : 1
root@TrabalhoSO-I:/home/root$cat arquivo

conteudo de arquivo continuacao do conteudo

root@TrabalhoSO-I:/home/root$format
root@TrabalhoSO-I:/home/root$ls
root@TrabalhoSO-I:/home/root$exit
CONSTRUÍDO COM SUCESSO (tempo total: 95 minutos 8 segundos)
```

O comando finaliza o terminal e dá fim a simulação.