Matheus Henrique Moreno da Silva

Douglas Donizeti de Castilho Braz Sistemas Operacionais II

31 de outubro de 2023

# Relatório do Trabalho

# Trabalho de simulação de arquivos em java

# Detalhes das Estruturas utilizadas para armazenamento do Sistema de Arquivos;

O Kernel é implementado na classe MyKernel.

implementadas neste Kernel:

* + ls(String parameters): Esta função lista os diretórios e arquivos existentes no sistema de arquivos. Os diretórios são armazenados em uma lista chamada d, e os arquivos em uma lista chamada a.
  + mkdir(String parameters): Cria um novo diretório com o nome especificado. Realiza verificações para garantir que o nome do diretório seja único.
  + cd(String parameters): Muda o diretório atual para o diretório especificado. Mantém o controle do diretório atual e a hierarquia de diretórios.
  + rmdir(String parameters): Remove um diretório especificado, desde que esteja vazio.

Realiza verificações para garantir que o diretório a ser removido existe e não contém arquivos ou subdiretórios.

* + cp(String parameters): Copia um arquivo ou diretório para um destino especificado.

Realiza verificações para garantir que a origem e o destino existam.

* + mv(String parameters): Move um arquivo ou diretório para um novo local. Garante que a origem e o destino sejam válidos.
  + rm(String parameters): Remove um arquivo ou diretório. Realiza verificações para garantir que o item a ser removido exista e está vazio (no caso de um diretório).
  + chmod(String parameters): Altera as permissões de um arquivo ou diretório. Verifica se o arquivo ou diretório existe e atualiza as permissões.
  + createfile(String parameters): Cria um novo arquivo no diretório atual ou especificado.

Realiza verificações para garantir que o nome do arquivo seja único.

* + cat(String parameters): Exibe o conteúdo de um arquivo especificado. Verifica se o arquivo existe e, se sim, retorna seu conteúdo.
  + batch(String parameters): Exporta a estrutura atual do sistema de arquivos para um arquivo de texto especificado.
  + dump(String parameters): Esta função é semelhante à função batch, mas exporta a estrutura do sistema de arquivos para um arquivo de texto diferente.
  + info(): Retorna informações sobre o aluno que implementou o Kernel, incluindo o nome, número de matrícula e a versão do Kernel.

# Dificuldade no desenvolvimento do trabalho. Teoria e/ou prática;

Desenvolver esse trabalho foi algo bastante desafiador. Projetar um sistema de arquivos foi bastante complexo; foi preciso muita imaginação e habilidades tanto práticas quanto teóricas.

Depois de muito tempo consegui transformar toda a teoria em código, tendo que ter maxima atencao nos detalhes.

# Complexidade das operações das chamadas de sistema implementadas (utilizando notação O);

ls (Listar Diretório):

* + Complexidade: O(n)
  + Justificativa: A operação "ls" envolve percorrer todos os diretórios e arquivos presentes no diretório atual. Portanto, a complexidade é linear em relação ao número de itens no diretório.

mkdir (Criar Diretório):

* + Complexidade: O(1)
  + Justificativa: A criação de um diretório é uma operação direta e não depende do número de diretórios existentes.

cd (Mudar Diretório):

* + Complexidade: O(n)
  + Justificativa: Para mudar de diretório, é necessário encontrar o diretório de destino, o que envolve uma pesquisa linear no diretório atual. Portanto, a

complexidade é linear em relação ao número de diretórios no diretório atual. rmdir (Remover Diretório):

* + Complexidade: O(n)
  + Justificativa: Remover um diretório requer encontrar o diretório de destino, o que envolve uma pesquisa linear no diretório atual. Além disso, se o diretório a ser

excluído não estiver vazio, pode ser necessário percorrer seus itens para

removê-los. Portanto, a complexidade é linear no número de diretórios no diretório atual.

cp (Copiar):

* + Complexidade: O(1)
  + Justificativa: A operação de cópia envolve apenas criar uma cópia de um arquivo ou diretório, sem depender do tamanho dos dados de entrada.

mv (Mover):

* + Complexidade: O(1)
  + Justificativa: Mover um arquivo ou diretório envolve apenas atualizar os ponteiros dos diretórios pai e filho, independentemente do tamanho dos dados.

rm (Remover Arquivo/Diretório):

* + Complexidade: O(1)
  + Justificativa: A remoção de um arquivo ou diretório envolve atualizar os ponteiros dos diretórios pai e filho, sem depender do tamanho dos dados.

chmod (Alterar Permissões):

* + Complexidade: O(1)
  + Justificativa: A alteração de permissões é uma operação direta e não depende do tamanho dos dados.

createfile (Criar Arquivo):

* + Complexidade: O(1)
  + Justificativa: A criação de um arquivo é uma operação direta e não depende do número de arquivos existentes.

cat (Concatenar Arquivo):

* + Complexidade: O(n)
  + Justificativa: A operação "cat" envolve ler o conteúdo de um arquivo, cuja complexidade é linear em relação ao tamanho do arquivo.

batch (Exportar Estrutura do Sistema de Arquivos):

* + Complexidade: O(n)
  + Justificativa: A exportação da estrutura do sistema de arquivos envolve listar todos os diretórios e arquivos presentes, resultando em uma complexidade linear em relação ao número de itens no sistema de arquivos.

dump (Exportar Estrutura do Sistema de Arquivos):

* + Complexidade: O(n)
  + Justificativa: A operação "dump" envolve listar todos os diretórios e arquivos presentes, resultando em uma complexidade linear em relação ao número de itens no sistema de arquivos.
  + HardDisk: complexidade O(N), onde N é o tamanho da memória secundaria em bytes, devido a alocação do array hardDisk.
  + getBlocos: complexidade é O(1) porque é uma operação direta de leitura de uma variavel de instancia.
  + inicializarMemoriaSecundaria:a complexidade é O(N), N é o tamanho da memoria secundaria
  + Na classe configuração temos os metodos get com complexidade O(1), pois só retornam o valor das variaveis estaticas.E os metodos set tambem sao O(1), pois só atualizam o valor de variaveis estaticas

# Sugestões, considerando as estruturas utilizadas, para a atualização que especifica um sistema de arquivos multiusuário;

Podemos começar implementando diretórios de usuários separados, para poder separar cada arquivo de usuários mantendo um diretório raiz para cada um deles.

# Sugestões, considerando as estruturas utilizadas, para a atualização que especifica um sistema de arquivos com acesso concorrente;

Pelo que já estudamos, podemos fazer isso com um escalonamento de acesso, se vários processos estiverem esperando para acessar um recurso bloqueado, um algoritmo de escalonamento justo pode ajudar a determinar a ordem de acesso.

# Comentários para as perguntas:

1. O sistema de arquivos implementado é eficiente?

O sistema de arquivos que desenvolvemos é eficiente, mas ate certo ponto, pois como uma estrutura estatica ficaria muito lento e complexo utilizar esse CMD em um sistema futuro. Ele foi projetado para minimizar o tempo que leva para acessar e manipular arquivos, tornando-o ágil e responsivo.

# O sistema de arquivos implementado é confiável?

Sim, mas nao 100% pois para tornar o sistema confiavel teriamos que implementar sistema de criptografia avançado, e o codigo implementado apenas trata de situações binarias.

# Quais testes foram executados para ‘garantir’ a confiabilidade do software?

Para ter certeza de que o nosso software é confiável, fizemos diferentes tipos de testes.

Alguns testes verificaram se as partes do software funcionam sozinhas, enquanto outros observaram como tudo funcionava junto. Também fizemos testes para ver como o sistema reagia a situações difíceis, como muita coisa acontecendo ao mesmo tempo ou até mesmo se o

computador desligasse repentinamente. Testamos muito tempo, diferentes situações e até o que acontece se alguém tentar fazer algo errado de propósito. Tudo isso nos deu confiança de que o nosso software funciona bem e é confiável.

# Você enxerga diferentes implementações de sistemas de arquivos para diferentes hardwares? Exemplifique.

Sim, diferentes hardwares podem exigir implementações específicas de sistemas de arquivos. Por exemplo, discos rígidos (HDDs) e unidades de estado sólido (SSDs) têm

características de acesso diferentes, o que pode afetar a forma como os dados são organizados e acessados.

CONCLUSÃO

Concluir este trabalho foi desafiador, mas recompensador. Aprendi muito sobre sistemas de arquivos e como traduzir a teoria em código prático. O sistema que criamos não é perfeito, mas para um primeiro contato com algo desse por podemos dizer que ele é eficiente e confiável.

Trabalhos citados

Livros:

"Sistemas de Arquivos" de Abraham Silberschatz, Peter Baer Galvin, e Greg Gagne. "Modern Operating Systems" de Andrew S. Tanenbaum.

"File System Forensic Analysis" de Brian Carrier