Relatório SO II – Sistema de Arquivos

a) Detalhes das Estruturas utilizadas:

- Item: classe-base para os tipos de estrutura que podem ser armazenadas no sistema, possui informações que são comuns para as estruturas:

→ Nome: String de no máximo 86 caracteres, acompanhado da extensão nos arquivos;

→ Estado: Número inteiro usado para marcar o tipo de estrutura de um determinado bloco, pode ser 0 (espaço livre), 1 (diretório) ou 2 (arquivo);

→ Pai: Número inteiro que marca o endereço do diretório onde o item em questão está armazenado;

→ Criação: String de exatamente 12 caracteres que mostra o momento de criação do item, composta por Ano (4 bits) – Mês (2 bits) – Dia (2 bits) – Hora (2 bits) – Minuto (2 bits);

→ Permissão: Número inteiro que indica as permissões dos usuários, composto por até 3 caracteres que irão indicar os 9 bits de permissão;

- Diretório: classe que implementa a classe Item, além das informações padrões, contém os seguintes conjuntos de endereços para salvar arquivos ou outros diretórios dentro de si:

→ mapDir: ArrayList de números inteiros, cada inteiro armazenado representa um outro diretório salvo no sistema e que pertence a este;

→ mapArq: ArrayList de números inteiros, cada inteiro armazenado representa um arquivo salvo no sistema e que pertence ao diretório;

- Arquivo: classe que implementa a classe Item, além das informações padrões, possui um espaço para salvar um texto de até 400 caracteres (variável conteúdo, do tipo String).

b) Dificuldades:

Na 1ª parte do trabalho, a principais dificuldades foram práticas e eram sobre navegar pelos diretórios corretamente ou manusear os diretórios em funções onde todas os itens de um diretório devem ser analisados (como nas funções “cp” e “chmod”), que necessitavam do uso de recursão.

Já na 2ª parte, além das funções recursivas, foi difícil manter a integridade dos itens de modo que pudessem ser escritos corretamente no HD, já que as dimensões escritas nele são fixas, e qualquer dado abreviado ou grande demais não era escrito adequadamente e causava uma exceção ou acabava não escrevendo algum diretório/arquivo copiado.

c) Complexidade das implementações:

- mkdir = O(n) - A estrutura mais complexa é o loop que executará sempre a quantidade n de diretórios presentes na String dos parâmetros, tanto os diretórios em si quanto os caracteres para navegação (“.” e “..”) são contabilizados;

- ls / cd / createfile / rmdir / cat = O(n) - Nessas funções, a estrutura mais complexa é a função auxiliar “findDir()” que executará um loop de magnitude igual a quantidade n de diretórios/caracteres de navegação inseridos nos parâmetros;

- rm / mv / chmod / cp = O(n^n) - Funções com recursividade, no caso onde devem manipular diretórios sua complexidade depende de chamar a função com todas as pastas descendentes de uma principal, independente do seu grau de parentesco;

- info = O(1) - Função para mostrar informações em variáveis, complexidade invariável;

- dump = O(n) - Sua complexidade varia de acordo com o número n de instruções que foram executadas desde que o SO foi iniciado e que serão escritas no documento;

- batch = O(n) - Sua complexidade varia de acordo com o número n de instruções que foram salvas dentro do documento no local indicado;

d) Sugestão para sistema multiusuário:

Poderia ser implementado de fato um sistema de permissão de usuários, utilizando as informações de permissão já existentes nos itens, junto com um sistema de hierarquia de usuários.

e) Sugestão para sistema com acesso concorrente:

Poderia ser implementado de fato um sistema de semáforo, para que o controle sobre um determinado arquivo fosse reservado a um usuário por vez.

f) Comentários:

I. O sistema de arquivos implementado é eficiente?

Sim, a complexidade da maioria das funções é relativamente baixa, e nas funções mais pesadas .

II. O sistema de arquivos implementado é confiável?

Sim, os itens são remontados por inteiro e ao serem modificados, diminuindo a chance de serem corrompidos ou escritos de maneira errada. Além disso, não é possível criar itens em blocos marcados pelo 1º byte (estado) diferente de 0, então não é possível sobrescrever itens.

III. Quais testes foram executados para 'garantir' a confiabilidade do software?

Além do teste disponibilizado pelo professor, foram feitos diversos testes em casos com entradas aleatórias, tanto para casos em que o erro já era previsto e barrado quanto em casos em que o sistema funciona normalmente.

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

Sim, por exemplo, um hardware de microcomputador, como de um relógio inteligente pode não precisar de um sistema de diretórios para organizar imagens armazenadas em si, pois sua memória é limitada, o que exige uma organização mais simples.