MAC0422 – Sistemas Operacionais – 1s2024
EP3 (Individual)
Data de entrega: 11/6/2024 até 8:00:00 da manhã

Prof. Daniel Macêdo Batista

1 Problema

A tarefa neste EP é implementar um simulador de sistemas de arquivos com vários comandos que simulem ações com arquivos dentro de tal sistema. O código deve ser escrito em C para ser executado no GNU/Linux. A interação com o usuário deve ser feita inteiramente no shell, sem interface gráfica, usando um prompt de comandos para enviar comandos para o simulador.

Os arquivos simulados no sistema de arquivos podem ser arquivos regulares ou diretórios e precisam ter os seguintes atributos armazenados:

- Nome
- Tamanho em bytes (menos no caso de diretórios)
- Instante de tempo em que foi criado
- Instante de tempo em que foi modificado pela última vez
- Instante de tempo em que foi acessado pela última vez

Além é claro dos dados de tais arquivos. No EP seu sistema de arquivos tratará apenas com arquivos em texto puro que serão copiados de algum local do sistema de arquivos real do computador.

2 Interação com o simulador

Quando executado na linha de comando (sem parâmetros) o simulador deve fornecer o prompt:

`{ep3}:`

Neste prompt os seguintes comandos precisam ser implementados:

- `monta arquivo`: monta o sistema de arquivos simulado que está no arquivo `arquivo`. Por exemplo, se o seu sistema de arquivos simulado estiver em `/tmp/unidadesimulada`, no seu computador, o comando será: `monta /tmp/unidadesimulada`. Se o `arquivo` já existir, o sistema de arquivos nele deve ser lido, a árvore de diretórios existente, completa, deve ser impressa na tela (inspire-se na saída do comando pstree), e a simulação deve continuar com o conteúdo já existente. Se o `arquivo` não existir, um novo sistema de arquivos simulado deve ser criado. Os comandos a seguir só podem ser executados caso o arquivo simulado tenha sido aberto com sucesso.

- `copia origem destino`: cria dentro do sistema de arquivos simulado uma cópia do arquivo `origem` que está em algum sistema de arquivos real dentro do seu computador. No sistema de arquivos simulado a cópia de `origem` será salva em `destino`. Tanto `origem` quanto `destino` devem ser informados com o caminho completo tanto dentro do sistema de arquivos real quanto no simulado[1]. Esse comando será executado apenas para copiar arquivos em texto puro.

- `criadir diretorio`: cria o diretório `diretorio`.

- `apagadir diretorio`: apaga o diretório `diretorio`. Se o diretório não estiver vazio, apaga tudo que estiver embaixo e avisa para o usuário tudo que foi apagado.

- `mostra arquivo`: mostra o conteúdo do arquivo `arquivo` na tela.

- `toca arquivo`: se o arquivo `arquivo` existir, atualiza o instante de tempo de acesso para o instante de tempo atual. Se não existir, cria um arquivo vazio.

- `apaga arquivo`: remove o arquivo `arquivo`.

- `lista diretorio`: lista os arquivos e diretórios que estejam imediatamente "embaixo" do diretório `diretorio`. Na listagem que será exibida, para todos os arquivos existentes, deverão ser exibidos: nome, tamanho em bytes, data de criação, data de última modificação e data de último acesso. Diretórios devem ser exibidos com alguma informação a mais que os diferencie como diretórios (por exemplo com uma / no final do nome).

- `atualizadb`: cria um "banco de dados", em memória, contendo a árvore inteira de arquivos e diretórios que existem no sistema de arquivos. Esse banco de dados só precisa guardar os caminhos e nomes de todos os arquivos e diretórios. Não é para guardar os conteúdos.

- `busca string`: busca no banco de dados criado com o comando anterior se há algum arquivo ou diretório que possua a `string` em qualquer parte do seu nome. Tudo o que for encontrado deve ser exibido na tela com os seus caminhos completos (apenas o nome, não o conteúdo). A busca pela string deve ser feita de forma simples. Não é necessário aplicar expressões regulares. Note que se um arquivo for apagado mas o comando `atualizadb` não for executado, o nome dele deve continuar sendo exibido caso possua a `string`.

- `status`: imprime na tela as seguintes informações do sistema de arquivos: quantidade de diretórios, quantidade de arquivos, espaço livre, espaço desperdiçado (considerando o espaço a mais necessário para cada arquivo ocupar exatamente múltiplos do tamanho de 1 bloco).

- `desmonta`: desmonta o sistema de arquivos.

- `sai`: sai do simulador.

Note que, uma vez finalizado o simulador, o arquivo que simula o sistema de arquivos não pode ser apagado. Ele deve continuar no estado em que foi deixado na última execução.

[1]Em todos os comandos a seguir, quando houver necessidade de especificar o nome de algum arquivo ou diretório, ele deve ser fornecido com o caminho completo

3 Requisitos

O simulador deve ser escrito em C. Programas escritos em outra linguagem ou utilizando alguma biblioteca extra para implementar os comandos de manipulação do sistema de arquivos terão nota zero.

O sistema de arquivos simulado equivale a uma única partição em um disco sem a necessidade de simular questões referentes a boot de um sistema operacional. Dessa forma, nenhum campo relativo a boot precisa ser simulado.

O sistema de arquivos simulado tem que suportar até 100MB, tanto de dados válidos quanto de metadados. Os tamanhos de blocos devem ser de 4KB. Se a execução de algum comando extrapolar o limite de 100MB, ela deve ser interrompida e uma mensagem de erro de espaço insuficiente deve ser exibida.

Com relação a diretórios, o sistema deve ter uma estrutura hierárquica começando no '/'. O caractere '/' deve ser o caractere separador entre diretórios.

Com relação ao armazenamento dos arquivos, deve ser feito usando FAT.

Com relação à implementação de diretórios, deve ser feita de modo que cada diretório seja uma lista com 1 entrada para cada arquivo dentro do diretório.

Com relação à gerência de espaço livre, deve ser feita usando bitmap.

Toda a simulação do sistema de arquivos e todas as operações nesse sistema devem ser implementadas do zero. EPs que utilizem bibliotecas ou similares que já implementem um sistema de arquivos ou os comandos terão nota ZERO.

4 Experimentos

Após finalizada a implementação do seu EP você deverá realizar experimentos que meçam o tempo médio (média de 30 execuções com intervalo de confiança e nível de confiança de 95%) das seguintes operações:

1. Cópia de um arquivo de 1MB no '/'
2. Cópia de um arquivo de 10MB no '/'
3. Cópia de um arquivo de 30MB no '/'
4. Remoção de um arquivo de 1MB no '/'
5. Remoção de um arquivo de 10MB no '/'
6. Remoção de um arquivo de 30MB no '/'
7. Remoção completa de um diretório "pai" com 30 níveis de hierarquia abaixo dele e sem arquivo regular em nenhum dos subdiretórios
8. Remoção completa de um diretório "pai" com 30 níveis de hierarquia abaixo dele e com centenas de arquivos regulares em todos os subdiretórios

Com o sistema de arquivos em 3 estados diferentes:

1. Sistema de arquivos vazio
2. Sistema de arquivos com 10MB ocupados
3. Sistema de arquivos com 50MB ocupados

5 Sobre a entrega

Deve ser entregue um arquivo .tar.gz contendo os itens listados abaixo. EPs que não contenham **todos** os itens abaixo **exatamente como pedido** terão nota ZERO e não serão corrigidos. **A depender da qualidade do conteúdo entregue**, mesmo que o EP seja entregue, **ele pode ser considerado como não entregue, o que mudará o cálculo da média final**:

- 1 único arquivo .c e 1 único arquivo .h;
- 1 arquivo LEIAME **em formato texto puro** explicando como compilar e executar o simulador, com os resultados dos experimentos explicados na Seção anterior e com o link para um vídeo mostrando a compilação e execução do código comprovando o funcionamento dos comandos. Para confirmar que de fato os comandos estão fazendo o que devem fazer, após executar uma sequência deles, desmonte o sistema de arquivos e rode o comando `hexdump` no arquivo, com as opções adequadas, para mostrar que o resultado é o esperado. A explicação do que estiver sendo exibido no vídeo pode ser feita com voz ou com legendas. O vídeo deve começar com a execução do comando md5sum no .c e no .h para confirmar que eles são exatamente os mesmos que foram entregues (O hash será comparado na correção). Logo, deixe para fazer o vídeo apenas quando você terminar de escrever o programa. Não altere nada dele depois que o vídeo for feito para evitar que o hash fique diferente. Vídeos que mostrem a execução de códigos com hashes que não batam com aqueles que foram entregues, serão desconsiderados (o mesmo vale para vídeos que tenham algum corte. Grave tudo de uma vez só);
- 1 único arquivo Makefile para gerar o executável.

O desempacotamento do arquivo .tar.gz deve produzir um diretório contendo os itens. O nome do diretório deve ser ep3-seu_nome. Por exemplo: ep3-mark_dean. Entregas que sejam tarbombs ou que, quando descompactadas, gerem o diretório com o nome errado perderão 1,0 ponto.

A entrega do .tar.gz deve ser feita através do e-Disciplinas.

O EP deve ser feito individualmente.

Obs.: não inclua no .tar.gz itens que não foram pedidos neste enunciado, como por exemplo, dotdirs como o .git, dotfiles como o .gitignore, saídas para diversas execuções, arquivos pré-compilados, etc.... A presença de conteúdos não solicitados no .tar.gz levarão a um desconto de 1,0 na nota final.