Universidade Federal do Rio Grande do Sul

INF01142 - Sistemas Operacionais IN

Prof. Alexandre da Silva Carissimi

Relatório do Trabalho Prático 2

T2FS

Arthur Giesel Vedana

Matrícula: 242238

16 de junho de 2016

Funcionamento das Funções

- identify2 – Funciona corretamente
- create2 – Funciona corretamente
- delete2 – Funciona corretamente
- open2 – Funciona corretamente
- close2 – Funciona corretamente
- read2 – Funciona corretamente
- write2 – Funciona corretamente
- seek2 – Funciona corretamente
- mkdir2 – Funciona corretamente
- rmdir2 – Funciona corretamente
- opendir2 – Funciona corretamente
- readdir2 – Funciona corretamente
- closedir2 – Funciona corretamente
- chdir2 – Funciona corretamente
- getcwd2 – Funciona corretamente

Testes

- **Teste1**

  Para testar um sistema de arquivos, decidi fazer um programa estilo “shell”. Obviamente, ele é extremamente simplificado e, ainda por cima, não possui muitas mensagens claras sobre os erros que podem ocorrer. Isso é, parcialmente, pois o T2FS retorna sempre o mesmo código de erro para qualquer coisa que aconteça, mas também é, em parte, por questão de simplicidade de implementação.

Ainda, a “shell” possui uma mistura de comandos tradicionais (cd, ls, mkdir, etc) com comandos mais baixo nível do que existem normalmente. Os comandos de baixo nível são os de leitura e escrita de arquivos, sendo simples cópias das funções disponíveis na T2Fs. Dessa forma, a escrita de novos dados no final de um arquivo deve ser feita com os seguintes comandos:

```
open file.txt
seek [handle] -1
write [handle] “Texto a ser inserido”
```

A shell foi implementada dessa maneira para melhor testar a implementação da biblioteca, permitindo chamadas diretas de suas funções. Observando os comandos anteriores, é possível ver que as operações em arquivos são feitas através de handles. Sempre que um arquivo é criado ou aberto, é informado o handle para a sua manipulação. A qualquer momento, é possível inserir o comando “viewopen” para ver a lista de arquivos abertos com seus nomes e handles.

Nos comandos mais tradicionais estão todos os associados à diretórios. É possível mudar o diretório corrente, listar os arquivos do diretório corrente, criar ou deletar diretórios usando a sintaxe tradicional de sistemas UNIX (ou pelo menos o que eu sei da sintaxe). Também foi implementada uma versão simplificada do programa “tree”, permitindo ver toda a árvore de arquivos abaixo do diretório corrente.

O programa possui uma lista dos comandos que aceita, e é possível ver essa lista escrevendo “help” (ou qualquer outra palavra que não seja um comando aceito). Isso mostrará a sintaxe de cada comando, e é possível entender o funcionamento da maioria só com essa informação. Se necessário, digitar o nome do comando sem parâmetros mostra informações mais detalhadas do que o comando faz (os únicos comandos sem parâmetros e que, portanto, não possuem informações detalhadas, são ls, tree e viewopen, que já foram explicados aqui).

Dificuldades de Implementação

Não sou um programador experiente em C, então muitas das minhas dificuldades foram causadas pela falta de conhecimento da linguagem e seus funcionamentos. Tive vários problemas com alocação de memória, que eu esquecia de liberar posteriormente, tendo de debugar com printfs espalhados pelo código. Também tive bastante dificuldade para lidar com as strings, já que elas são tratadas de forma muito mais baixo nível do que nas linguagens que estou acostumado a programar.

Também tive dificuldade para verificar o progresso que estava fazendo. No começo, passei um bom tempo criando estruturas e funções para imprimir o estado atual do disco, já que é muito difícil analisar os bytes crus num leitor binário. Depois de criar essas funções auxiliares, pude começar a realmente ver se as funções estavam tendo os resultados esperados, facilitando muito o processo de debugar.

Tive bastante trabalho, não necessariamente dificuldade, ao tentar lidar com a leitura e escrita de arquivos, principalmente com a parte de dupla indireção. Acabei criando duas funções auxiliares, uma

para descobrir qual setor lidava com um determinado offset de um arquivo (por exemplo, o offset 1024 de um arquivo retornaria o setor do dataPtr[1]) e outra para salvar que um determinado bloco foi adicionado a um arquivo. A primeira ficou relativamente compacta, com 74 linhas. A segunda, com 155 linhas, foi mais difícil de fazer. Além de precisar verificar em qual bloco de índices o novo ponteiro deveria ser adicionado, era necessário verificar se o próprio bloco de índices existia, carregando-o e limpando seu conteúdo caso contrário.

Tive bastante duplicação de código para fazer essas duas funções, já que o tratamento do bloco de indireção simples é exatamente igual ao tratamento do segundo nível na indireção dupla. Mesmo sabendo dessa duplicação, não consegui eliminá-la de modo eficiente, e acabei deixando um código um pouco menos legível como consequência.