Universidade Federal do Rio Grande do Sul
Instituto de Informática – Informática Aplicada
Sistemas Operacionais I – 2014/I
1 de julho de 2014

IMPLEMENTAÇÃO DE UM SISTEMA DE ARQUIVOS

Arthur Ribeiro – 207834
Gabriel St-Denis – 247170
Ronaldo Dall'Agnol Veiga – 209823

1. ***Sem alterar a quantidade de ponteiros de alocação indexada, quais outros fatores influenciam no maior tamanho de arquivo T2FS possível? Como esse fatores influenciam nesse tamanho?***

   O tamanho dos blocos influencia no maior tamanho de arquivo T2FS possível porquê com blocos de tamanho maior, mais dados e mais ponteiros em blocos de índices podem ser armazenados por bloco, assim, um arquivo pode ser de tamanho maior.

2. ***Supondo que você desejasse melhorar o T2FS, permitindo a criação de vínculos estritos (hardlinks). Que alterações seriam necessárias no T2FS? Há necessidade da criação de novas funções? Se sim, quais? Se não, porque não.***

   Seria necessário definir um novo tipo de arquivo (hardlinks) para os vínculos estritos. Então, um arquivo desse tipo teria o ponteiro para o arquivo desejado. Seria necessário de criar uma função para a criação desse novo tipo de arquivo e uma outra função para acessar as informações nele contidas.

3. ***As estruturas de controle do T2FS contêm informações que permitem verificar a consistência de alguns de seus elementos. Isso é possível graças a um nível de redundância de informação (por exemplo, no registro de arquivo, nas entradas do diretório, o número total de blocos usados por um arquivo e o tamanho do arquivo – em bytes – permitem uma verificação). Identifique quais outros elementos são redundantes e discuta como seria possível usar essa redundância para aumentar a confiabilidade do T2FS.***

   Tem uma redundância entre o número de blocos usados e o número de ponteiros validos. Essa redundância poderia ser usada num caso onde ocorre um erro no disco e que a informação sobre o número de blocos é corrupta, portanto, seria ainda possível de saber o número de blocos usados a partir do número de ponteiros validos. E vice-versa, se dos ponteiros são perdidos num erro do disco, ainda é possível de saber o número de ponteiros validos com o número de blocos usados.

   Uma redundância que poderia também ser implementada é a duplicação dos arquivos Bitmap e dos ponteiros de registros. Assim, no caso que um erro do disco ocorre e que um arquivo Bitmap é perdido, fica a cópia desse arquivo Bitmap.

4. ***Como você implementou a atribuição dos identificadores de arquivos (file handler) pelas funções t2fs_create e t2fs_open? Discuta a questão da reutilização dos mesmos.***

   A cada vez que um arquivo é criado, um inteiro armazenado numa variável global estática é incrementado, assim, o identificador desse arquivo é definido igual a esse inteiro incrementado. A cada vez que um arquivo é aberto, uma struct FileHandle é inserida numa lista encadeada de arquivos abertos, esse FileHandle contem o identificador do arquivo e outra informação sobre o arquivo.

5. ***Como você implementou a gerência do contador de posição (current pointer) usado pela função t2fs_seek?***

   Uma das informações sobre o arquivo que é na struct FileHandle é para a posição atual do ponteiro no arquivo. Essa informação é modificada cada vez que é realizada uma escrita ou leitura no arquivo para seja a posição atual do ponteiro no arquivo. Essa informação é também modificada quando é chamada a função t2fs_seek, ao momento da chamada da função, a essa informação é adicionado o valor do parâmetro offset da função.

6. ***A escrita em um arquivo (realizada pela função t2fs_write) requer uma sequência de leituras e escritas de blocos de dados e de blocos de controle. Qual é a sequência usada por essa função? Se essa sequência for interrompida (por falta de energia, por exemplo) entre duas operações de escrita de bloco, qual será o efeito na consistência dos dados no disco? É possível projetar uma sequência de escritas no disco que minimize a eventual perda de dados?***

   Para uma escrita em fim de arquivo, a sequência a seguinte:

   1. Reservar os blocos necessários no Bitmap para escrever os dados;
   2. Atualizar os ponteiros do registro descrevendo o arquivo;
   3. Escrever os dados nos blocos reservados.

   No caso que ocorre uma interrupção da escrita entre as etapas 1 e 2, da memória é perdida porquê ela é alocada mas não usada. No caso que a interrupção ocorre entre as etapas 2 e 3, dado corrupto fica no arquivo, assim o arquivo é corrupto.

   Para minimizar os dados perdidos no caso que ocorre uma interrupção, além de reservar todos os blocos antes de começar a escrever a dentro, é melhor reservar um bloco, escrever a dentro e quando ele esta cheio, reservar um novo bloco e escrever outros dados a dentro. Assim, a interrupção causa menus perdida de dados mas o desempenho do sistema é um pouco mais fraco.

7. ***Algumas estruturas gravadas no disco são mais facilmente manipuláveis se estiverem na memória principal (como se fosse uma cache). Por outro lado, isso aumenta a possibilidade de perda de dados, pois as informações existentes nessa cache e que não foram escritas no disco, podem ser perdidas, caso ocorra alguma interrupção de operação do sistema. Quais informações do disco você está mantendo (e gerenciando) na memória principal e porque você as escolheu? Qual a política que você usou para***

decidir quando escrevê-las no disco?

O arquivo Bitmap e seu endereço está mantendo na memória principal porquê ele é acessado cada vez que uma escrita ou leitura dum arquivo ocorre. Aceder ao arquivo Bitmap no disco a cada vez que ele é acessado seria bem ineficiente. A escrita no arquivo Bitmap no disco é feita somente ao fim da escrita na memoria principal. Assim, se uma interrupção do sistema ocorre durante a escrita, os dados modificados no Bitmap na memória principal e ainda não no disco são perdidos.

8. ***Todas as funções implementadas funcionam corretamente? Relate, para cada uma das funções desenvolvidas, como elas foram testadas?***

   Todas as funções são implementadas, mas, não foram testadas por falta de tempo. Então, é bem provável que algumas não funcionam bem.

9. ***Relate as suas maiores dificuldades no desenvolvimento deste trabalho e como elas foram contornadas.***

   Encontramos algumas dificuldades na implementação da estrutura em árvore. Foi bem difícil de entender como funcionam os ponteiros dum arquivo de tipo diretório. Foi claro para nós que um arquivo ponta para o arquivo antes e o arquivo depois dele num diretório. Mas, a implementação da lógica onde um arquivo de tipo diretório deve pontar para os arquivos a dentro dele foi difícil. Também, implementar as funções para gerar o Bitmap foi difícil porquê não tem de funções para tratar os bits numa variável em C.