

# Sistemas Operacionais

## Exercício Programa 4

### Sistema de Arquivos

## Data de entrega: 04/07/2014

#### AVISO

Para este projeto você vai modificar o sistema de arquivos. CUIDADO com as modificações, se você fizer besteira, pode danificar seu sistema de arquivos irremediavelmente. Assim, tenha certeza de fazer backups constantes e testar incrementalmente o que você está fazendo. **Faça a parte 2 antes da parte 1!!!** Programas apenas com a parte I não serão corrigidos. Programas apenas com parte II terão crédito parcial. Cuidado, a estrutura do sistema de arquivos do MINIX mudou muito a partir da 3.1.6, passando a ter componentes separados. Vocês devem, por simplicidade e por apoio do livro, usar 3.1.0.

#### ***Parte I (7 pontos): Implementando "arquivos imediatos" em Minix:***

Nesta fase você deve implementar suporte a "arquivos imediatos", ou seja **arquivos cujo conteúdo será armazenado no próprio i-node**. Esta é uma maneira de otimizar o acesso a arquivos bem pequenos, que passarão a exigir apenas um acesso a disco para ter seu conteúdo carregado na memória.

Como vocês sabem, cada arquivo em Minix é descrito por uma estrutura chamada i-node, que contém ponteiros para localizar os blocos de disco onde a informação do arquivo é guardada. Porém, para arquivos pequenos, isso pode gerar um grande overhead de espaço e de tempo para o acesso inicial

Entre outras informações o i-node registra o endereço dos primeiros blocos do arquivo em uma área interna (veja a figura no final deste enunciado). Estes ponteiros podem indicar diretamente um bloco do disco onde são armazenadas informações ou os blocos iniciais das estruturas de indireto, indireto duplo e indireto triplo.

Podemos utilizar *arquivos imediatos* para tornar armazenamento de pequenos arquivos mais eficiente e para minimizar fragmentação interna. Como dissemos, um arquivo imediato terá seu conteúdo armazenado diretamente no i-node, **na área reservada para os ponteiros de blocos. No MINIX, 40 bytes são reservados para os ponteiros de blocos e você deverá utilizar os 40 bytes como armazenamento para seu arquivo imediato.**

Sempre que um arquivo novo for criado (com tamanho 0), você deve criá-lo como arquivo imediato e armazenar as informações no próprio i-node. **Quando o tamanho do arquivo exceder 32 bytes, ele deve ser transformado em um arquivo regular.** Isso quer dizer que quando isso acontecer, um novo bloco deve ser

alocado, a informação guardada no i-node copiada, e o endereço do novo bloco deve ser armazenado na primeira entrada de blocos do i-node. Para simplificar um pouco, NÃO vamos exigir que arquivos que tenham seu tamanho reduzido a menos de 32bytes sejam transformados em arquivos imediatos. Em particular isso quer dizer que não será necessário alterar nenhum dos arquivos atuais do sistema.

Isso envolve alguns passos importantes. Primeiro você precisa sinalizar de alguma maneira no i-node que um arquivo é imediato (lembre-se que arquivos normais que têm seu tamanho reduzido, continuam como arquivos normais). Para isso precisará de um campo extra (veja se existe algum espaço disponível). Em segundo lugar, pense nas operações que podem ser afetadas quando um arquivo é imediato, e modifique-as adequadamente. Alguns requisitos

1. Quando um arquivo for criado, ele deve ser imediato
2. Sempre que um arquivo for ser eliminado (com uma operação de deleção ou quando os links para o i-node chegam a zero), seu código NÃO deve tentar livrar blocos (já que não há nenhum alocado).
3. Quando uma operação de leitura for realizada, você deve retirar os dados do próprio i-node
4. Quando uma operação de escrita for realizada, você deve verificar se isso tornará o arquivo grande demais para ser imediato. Neste caso você deve alocar um novo bloco no disco, copiar os dados que estão guardados no i-node e atualizar a área de ponteiros do i-node (cuidado, inclusive as entradas que não apontarão para nenhum bloco).
5. Veja nos arquivos do sistema de arquivos para entender como as operações são realizadas atualmente e para localizar o código que precisa ser mudado. Reutilize funções sempre que possível.
6. FAÇA TUDO AOS POUCOS, testando a cada passo. Tentar fazer tudo de uma vez tornará impossível localizar os problemas.

### **Parte 2 (3 pontos): Verificação dos arquivos (esta parte deve ser feita antes, para facilitar testar seu sistema)**

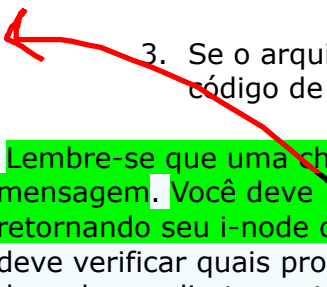
Nesta parte você irá implementar uma nova funcionalidade importante para fazer a validação da outra parte. Você deverá acrescentar uma chamada de sistema que permita ao usuário monitorar o uso de arquivos no sistema. Esta chamada deverá se chamada `lsr (listar recursos)`, com a seguinte interface

`Int lsr(char* caminho)`

Onde, *caminho* contém uma string com o caminho absoluto (ex. `/home/eu/arq.txt`) ou relativo (ex. `../outro/arq2.txt`). Quando a chamada for feita o sistema deve exibir, na entrada padrão,

1. A lista dos ids de todos os processos que estão com o arquivo aberto
2. A lista dos blocos do disco que abrigam o conteúdo do arquivo Se o arquivo for imediato, ou se estiver vazio, indicar isso na saída


verificar o que é isso

- 
3. Se o arquivo descrito não existir, a chamada deve retornar um código de erro ao usuário e imprimir uma mensagem

Lembre-se que uma chamada de sistemas é sempre convertida para uma mensagem. Você deve primeiro resolver o caminho com o nome do arquivo retornando seu i-node ou retornando um erro se ele não existir. Caso exista, deve verificar quais processos estão com este arquivo aberto e imprimir o PID de cada um diretamente do VFS (não precisa retornar a lista ao processo que chamou lsr). Em seguida você deve encontrar todos os blocos de disco alocados a este arquivo. Comece com os ponteiros no i-node (se houver), em seguida percorrer os ponteiros associados aos esquemas indiretos (indireto simples e indireto duplo). Você deve imprimir os números dos blocos, NA ORDEM QUE SAO UTILIZADOS PELO ARQUIVO, separados por espaços

Nota final: Utilize, na medida do Possível, funções já existentes no minix, como funções para encontrar um i-node dado um caminho, ou para empacotar uma string em uma mensagem (sugestão, siga a chamada de sistema de abertura de arquivos).

### Detalhes Administrativos

1. O trabalho deve ser executado em duplas. Como regra cada EP gerará uma nova imagem do sistema, mas as imagens serão sempre construídas em cima da imagem anterior, assim recomendamos que o grupo seja mantido no semestre. Mudanças devem comunicadas ao professor
  2. Trabalhos atrasados terão uma penalidade de 10% da nota por dia de atraso.
  3. Entrega: você deve produzir uma imagem .ova e um documento PDF.
    - a. A imagem .ova deve conter os novos arquivos fontes adicionados à biblioteca do sistema para implementar a nova chamada.e o sistema com as modificações pedidas.
    - b. O documento pdf deve conter um relatório sucinto do que foi feito, incluindo a localização dos arquivos novos e dos arquivos modificados.
    - c. Você deve incluir em sua imagem os arquivos fonte e executável dos programas utilizados para teste e descrever no relatório como estes foram realizados.
    - d. Indique também em seu relatório o tamanho da memória utilizada no emulador.
    - e. Nos arquivos modificados o código novo deve estar bem ressaltado com linhas de comentário de '#' antes de depois das linhas modificadas, Ex:
- 

```
/*#####*/
```

<codigo modificado>

```
/*#####*/
```

