Acesso ao disco

Este projeto faz parte do PPOS v2.

Este projeto tem por objetivo implementar operações de entrada/saída (leitura e escrita) de blocos de dados sobre um disco rígido virtual. A execução dessas operações estará a cargo de uma tarefa **gerente de disco**, que cumpre a função de *driver* (piloto) de acesso ao disco.

O disco virtual

O disco virtual simula o comportamento lógico e temporal de um disco rígido real, com as seguintes características:

- O conteúdo do disco virtual é mapeado em um arquivo UNIX com nome default `disk.dat`. O conteúdo do disco virtual é persistente, ou seja, é mantido de uma execução para outra.
- O disco contém N blocos de mesmo tamanho. O número de blocos do disco dependerá do tamanho do arquivo subjacente no sistema real.
- As operações de leitura/escrita são feitas sempre com um bloco de cada vez. Não é possível ler ou escrever bytes isolados, parte de um bloco, ou vários blocos ao mesmo tempo.
- Os pedidos de leitura/escrita de blocos são **assíncronos**, ou seja, apenas registram a operação solicitada e retornam imediatamente, sem bloquear a tarefa. A finalização de cada operação de leitura/escrita é indicada mais tarde pelo disco através de uma interrupção virtual, que deve ser capturada e tratada.
- O disco só trata uma leitura ou escrita por vez. Enquanto o disco está tratando uma solicitação, ele fica em um estado ocupado (*busy*); tentativas de acesso a um disco ocupado retornam um código de erro.
- O tempo de resposta do disco é proporcional à distância entre a posição atual da cabeça de leitura do disco e a posição da operação solicitada. Inicialmente a cabeça de leitura está posicionada sobre o bloco inicial (zero).

O código que simula o disco está em `hardware/disk.c` e sua interface de acesso está definida em `hardware/disk.h`; estes arquivos **não devem ser modificados**.

O acesso ao disco deve feito **somente** através das definições presentes em `disk.h`. O código presente em `disk.c` implementa o comportamento interno do disco virtual e pode ser abstraído.

Interface de acesso ao disco

O acesso ao disco pelas tarefas é feito através das funções definidas em `kernel/block.h` e implementadas (pelo aluno) em `kernel/block.c`, usando a interface oferecida pelo disco virtual em `hardware/disk.h`:

Iniciar o subsistema

```
void block_init(char *disk_image)
```

Esta função, chamada na inicialização do PPOS (em `kernel/ppos.c`) inicia o subsistema de acesso ao disco. Ela recebe como parâmetro o nome do arquivo que armazena o conteúdo do disco virtual.

Informações sobre o disco

```
int block_size()

int block_blocks()
```

Estas duas funções retornam respectivamente o tamanho de cada bloco do disco virtual (em bytes) e o número de blocos do mesmo.

Leitura e escrita de blocos

As tarefas podem ler e escrever blocos de dados no disco virtual através das seguintes chamadas:

```
int block_read(int block, void* buffer)

int block_write(int block, void* buffer)
```

Parâmetros:

- `block`: número do bloco a ler ou escrever no disco (entre 0 e *número de blocos* - 1);
- `buffer`: endereço do *buffer* com os dados a escrever no disco, ou onde devem ser colocados os dados lidos do disco; esse *buffer* deve ter capacidade para `block_size` bytes.
- retorno: 0 em caso de sucesso ou -1 em caso de erro.

Estas funções são **bloqueantes**: cada tarefa que solicita uma leitura/escrita no disco **deve ser suspensa** até que a operação solicitada seja completada, liberando o processador para outras tarefas.

As requisições de acesso ao disco são mantidas em uma fila específica, gerenciada por uma tarefa "gerente de disco". Essa tarefa submete as requisições ao disco e trata as interrupções geradas pelo mesmo ao concluir cada operação.

Neste projeto, as solicitações de leitura/escrita devem ser atendidas na ordem em que foram feitas, de acordo com a política de escalonamento de disco FCFS (*First Come, First Served*).

Tarefa

Este projeto consiste em implementar no arquivo `kernel/block.c`:

- Uma tarefa gerente do disco;
- Uma função para tratar as interrupções virtuais geradas pelo disco (vide `hardware/cpu.h`), acordando a tarefa gerente de disco quando necessário;
- Uma fila genérica de pedidos de acesso ao disco; cada pedido na fila indica a tarefa solicitante, o tipo de pedido (leitura ou escrita), o bloco desejado e o endereço do *buffer* de dados;
- As funções de leitura/escrita oferecidas às tarefas (`block_read` e `block_write`);
- As demais funções definidas em `kernel/block.h`.

Arquivos

Os seguintes arquivos são relevantes para este projeto:

- `hardware/disk.h`: interface de acesso ao disco virtual.

- `hardware/cpu.h`: funções de tratamento de IRQs.
- `hardware/disk.dat`: conteúdo inicial do disco virtual, que tem 256 blocos de 64 bytes cada (b_0, b_1, ..., b_{255}), totalizando 16.384 bytes. Para facilitar a visualização, o conteúdo inicial de cada bloco é o número do bloco e alguns caracteres de enchimento para completar os 64 bytes.
- `kernel/block.h`: interface de do subsistema de blocos.
- `kernel/block.c`: implementação do subsistema de blocos.
- `test/pingpong-disco.c`: programa de teste simples: uma tarefa única, que lê os blocos do disco em sequência e imprime seu conteúdo na tela. Em seguida, ela escreve nos blocos em sequência, com caracteres aleatórios.
- `test/pingpong-disco-stress.c`: várias tarefas leem e escrevem no disco simultaneamente, com o objetivo de inverter a ordem dos blocos do mesmo (b_{255}, b_{254}, ..., b_0). O conteúdo final do disco deve ser igual ao contido em `hardware/disk-final.dat`.
- `test/pingpong-disco*.txt`: saídas esperadas dos programas de teste.

Sugestão de implementação

As funções `block_read` e `block_write` devem seguir +/- o seguinte comportamento:

```
block_read / block_write
{
    monta um pedido de acesso ao disco
    insere o pedido na fila de pedidos
    acorda o gerente de disco
    suspende a tarefa atual
}
```

A função de tratamento da interrupção do disco deve ser acionada a cada ocorrência da IRQ do disco; seu comportamento é bem simples:

```
trata_irq_disco
{
    ocorreu_irq = true
    acorda o gerente de disco
}
```

A tarefa *gerente de disco* é responsável por tratar os pedidos de leitura/escrita das tarefas e as operações concluídas pelo disco. Ela deve ser acordada (com `task_awake`) sempre que alguma tarefa pedir uma operação de leitura/escrita no disco ou quando o disco gerar uma IRQ informando que a última operação solicitada foi concluída.

Ela é uma tarefa de sistema, similar ao *dispatcher*, e tem o seguinte comportamento:

```
disk_manager_body
{
    repetir
    {
        se ocorreu_irq
        {
            ocorreu_irq = false
            acorda a tarefa cujo pedido foi atendido
        }

        se o disco estiver livre e houver pedidos na fila
        {
```

```
            retira um pedido da fila (usando FCFS)
            submete o pedido ao disco
        }

        suspende a tarefa atual
    }
}
```

A fila de pedidos de acesso ao disco é uma estrutura de dados compartilhada, que pode ser acessada de forma concorrente pelo gerente de disco e pelas funções de leitura e escrita. Há risco de **condições de disputa** em seu acesso, portanto ela deve ser acessada com **exclusão mútua** (usando semáforo) para garantir sua integridade.

Outras informações

- Duração estimada: 8 horas.
- Dependências:
  - Tarefas cooperativas
  - Despachante de tarefas
  - Preempção por Tempo
  - Tarefas que esperam
  - Tarefas que dormem
  - Spinlocks e semáforos