Gerente de disco

Video deste projeto

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 estarão a cargo de um **gerente de disco**, que cumpre a função de *driver* de acesso ao disco.

Interface de acesso ao disco

A tarefa principal (`main`) inicializa o gerente/driver de disco através da seguinte chamada:

```
int disk_mgr_init (&num_blocks, &block_size) ;
```

Ao retornar da chamada, a variável `num_blocks` contém o número de blocos do disco inicializado, enquanto a variável `block_size` contém o tamanho de cada bloco do disco, em bytes. Essa chamada retorna 0 em caso de sucesso ou -1 em caso de erro.

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

```
int disk_block_read  (int block, void* buffer) ;
int disk_block_write (int block, void* buffer) ;
```

- `block`: posição (número do bloco) a ler ou escrever no disco (deve estar entre 0 e `numblocks-1`);
- `buffer`: endereço dos 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.

Cada tarefa que solicita uma operação de leitura/escrita no disco **deve ser suspensa** até que a operação solicitada seja completada. As tarefas suspensas devem ficar em uma fila de espera associada ao disco. As solicitações de leitura/escrita presentes nessa fila devem ser atendidas na ordem em que foram geradas, de acordo com a política de escalonamento de disco FCFS (*First Come, First Served*).

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 no diretório corrente da máquina real, com nome default `disk.dat`. O conteúdo do disco virtual 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.
- Como em um disco 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 feitos ao disco são assíncronos, ou seja, apenas registram a operação solicitada, sem bloquear a chamada. A finalização de cada operação de leitura/escrita é indicada mais tarde pelo disco através de um sinal UNIX `SIGUSR1`, que deve ser capturado e tratado.
- O disco só trata uma leitura/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 `disk.c` e sua interface de acesso está definida em `disk.h`; estes arquivos **não devem ser modificados**.

O acesso ao disco deve feito **somente** através das definições presentes em `disk.h`. As definições presentes em `disk.c` implementam (simulam) o comportamento interno do disco e por isso **não devem ser usadas** em seu código.

Atenção: o arquivo `disk.c` depende da biblioteca POSIX de tempo real (`-lrt`):

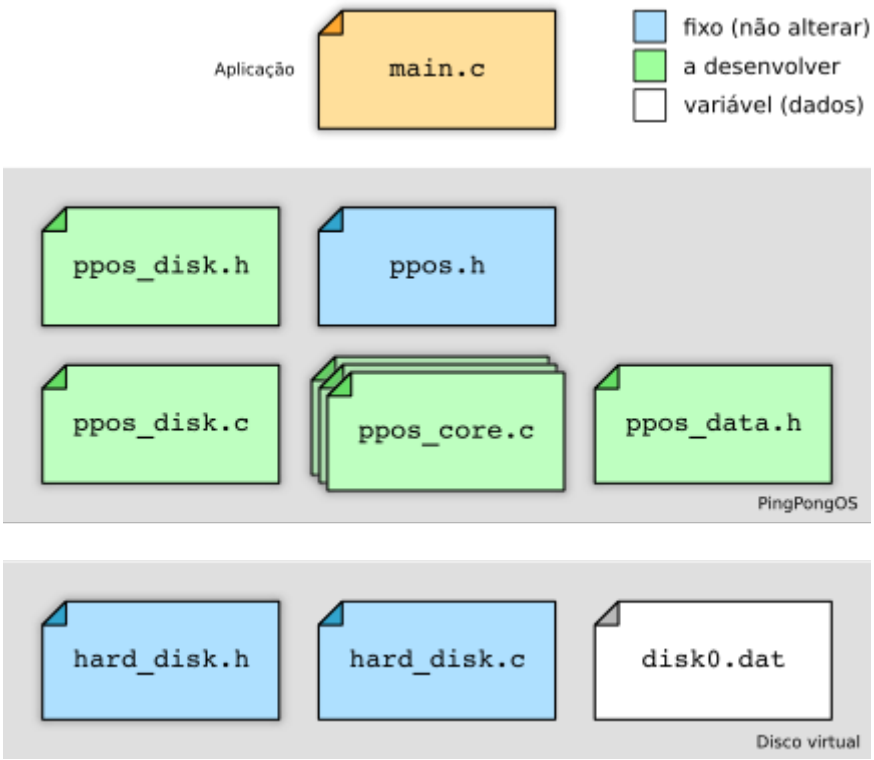
```
cc -Wall queue.c ppos_core.c ppos_disk.c disk.c pingpong-disco.c -lrt
```

Tarefa

A gerência das operações de entrada/saída em disco consiste em implementar:

- Uma tarefa gerenciadora do disco;
- Uma função para tratar os sinais `SIGUSR1` gerados pelo disco, que acorda a tarefa gerenciadora de disco quando necessário;
- Uma fila de pedidos de acesso ao disco; cada pedido indica a tarefa solicitante, o tipo de pedido (leitura ou escrita), o bloco desejado e o endereço do buffer de dados;
- As funções de acesso ao disco oferecidas às tarefas (`disk_mgr_init`, `disk_block_read` e `disk_block_write`).

A implementação do gerenciamento de disco deve ficar no arquivo `ppos_disk.c`, enquanto sua interface fica no arquivo ppos_disk.h (fornecido, a completar). A figura a seguir mostra a estrutura geral do código:



Testes

Sua implementação deverá funcionar com estes arquivos de teste:

- 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 o bloco.
- pingpong-disco1.c: 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. A saída deve ser similar a este exemplo.
- pingpong-disco2.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 a este e a saída deve ser similar a este exemplo.

Sugestão de implementação

A tarefa *gerente de disco* é responsável por tratar os pedidos de leitura/escrita das tarefas e os sinais gerados pelo disco. Ela é uma tarefa de sistema, similar ao *dispatcher*, e tem +/- o seguinte comportamento:

```
void diskDriverBody (void * args)
{
   while (true)
   {
      // obtém o semáforo de acesso ao disco

      // se foi acordado devido a um sinal do disco
      if (disco gerou um sinal)
      {
         // acorda a tarefa cujo pedido foi atendido
      }

      // se o disco estiver livre e houver pedidos de E/S na fila
      if (disco_livre && (fila_disco != NULL))
      {
         // escolhe na fila o pedido a ser atendido, usando FCFS
         // solicita ao disco a operação de E/S, usando disk_cmd()
      }

      // libera o semáforo de acesso ao disco

      // suspende a tarefa corrente (retorna ao dispatcher)
   }
}
```

A tarefa gerente de disco deve ser acordada (voltar à fila de tarefas prontas) sempre que:

- alguma tarefa pedir uma operação de leitura/escrita no disco; ou
- o disco gerar um sinal indicando que a última operação solicitada foi concluída.

Dessa forma, as funções `disk_block_read` e `disk_block_write` devem seguir +/- o seguinte comportamento:

```
disk_block_read (block, &buffer)
```

```
{
   // obtém o semáforo de acesso ao disco

   // inclui o pedido na fila_disco

   if (gerente de disco está dormindo)
   {
      // acorda o gerente de disco (põe ele na fila de prontas)
   }

   // libera semáforo de acesso ao disco

   // suspende a tarefa corrente (retorna ao dispatcher)
}
```

Outras informações

- Duração estimada: 8 horas.
- Dependências:
  - Gestão de Tarefas
  - Dispatcher
  - Preempção por Tempo
  - Tarefas suspensas
  - Tarefas dormindo
  - Semáforos