Construção de Sistemas Operacionais - Trabalho 2

Prof. Sérgio Johann Filho

Objetivos

- Expandir o conhecimento sobre construção e suporte de distribuições do sistema operacional Linux.
- Compreensão das estruturas de dados e interfaces internas do kernel Linux.
- Projeto e implementação de *device drivers*.

Descrição

Este trabalho consiste no projeto implementação de um módulo carregável de kernel (KLM - *kernel loadable module*). Esse módulo terá como objetivo implementar um mecanismo para distribuição de mensagens entre processos utilizando um sistema de filas circulares (acessível em */dev/mq*). As filas poderão ser acessadas por meio de uma aplicação que faz uso da API de acesso ao driver. Devem existir múltiplos processos em espaço de usuário, e cada um poderá realizar operações de leitura ou escrita no driver. Com isso, o driver será responsável por fornecer ou armazenar mensagens de processos específicos.

As operações de escrita deverão enfileirar mensagens em uma estrutura de dados gerenciada no próprio driver (em espaço de kernel), de tal forma que um processo possa consumir posteriormente mensagens enviadas ao driver e a ele endereçadas. A estrutura de dados será, dessa forma, uma lista de filas de mensagens, onde cada elemento da lista será um ponteiro para uma fila circulare de mensagens para um processo específico. O sistema de filas circulares permite o armazenamento das últimas mensagens enviadas a um determinado processo.

Para o desenvolvimento do módulo carregável, deverão ser utilizadas as funções *kmalloc()* e *kfree()* bem como a API de listas como estruturas de dados disponíveis em nível de kernel. É importante que a gerência de memória seja implementada corretamente, evitando o travamento do sistema e vazamentos de memória.

Funcionamento

O módulo deverá ser carregado da forma ilustrada abaixo, sendo necessário informar os seguintes parâmetros ao módulo: o número máximo de processos que podem acessar o sistema de filas, o número máximo de mensagens enfileiradas (por processo) e o tamanho máximo de cada mensagem (em bytes). Importante ressaltar que, caso um processo receba mais mensagens do que o número máximo, as mensagens mais antigas deverão ser descartadas.

```
$ modprobe mq_driver 20 8 128
```

Após a sua carga, o módulo (driver) deverá atender solicitações de processos em nível de usuário, sendo o seu comportamento implementado em uma aplicação. Antes de poder enviar uma mensagem para outros processos, um processo deve registrar-se, associando a si um nome por meio de uma operação de escrita (o módulo do kernel deve associar o *pid* do processo que realizou a operação).

```
/reg processo1
```

Após o registro, um processo (aplicação no espaço de usuário) pode enviar dados a outro processo por meio de uma operação de escrita. Por exemplo, para enviar uma mensagem a um processo com o nome "processo1", um outro processo já registrado pode realizar uma operação de escrita em */dev/mq* com uma mensagem no seguinte formato, a ser enviado como uma string para o módulo:

```
/processo1 tudo bem por ai?
```

Diversas solicitações desse tipo podem ser enviadas por diferentes processos, sendo que o driver deverá manter uma lista de processos que são destino das mensagens (processos registrados), e associado a cada elemento dessa lista deverá existir outra lista (fila circular), contendo as últimas mensagens destinadas a este processo. Posteriormente, processos que realizarem operações de leitura receberão dados vindos do driver, que deverá retornar (em ordem) as mensagens endereçadas a esse processo, uma a uma, por operações sucessivas de leitura (não devem ser retornadas mensagens destinadas a outros processos). Cada operação de leitura efetuada por um processo removerá uma única mensagem da sua fila de mensagens. O módulo pode identificar um processo leitor por seu *pid*, que estará registrado e associado ao seu nome. Ao realizar uma operação de leitura, um processo poderá não receber mensagem alguma (caso sua fila esteja vazia), ou uma mensagem que estará na cabeça da fila associada ao seu nome.

Mensagens destinadas à processos que não estão registrados deverão ser descartadas. Nesse caso, é importante que o driver emita um alerta. Além disso, o driver deve suportar o envio de mensagens a todos os outros processos registrados (deverá ser colocada uma cópia de uma mensagem em cada fila de cada processo) usando-se o seguinte formato, enviado como uma string para o módulo:

```
/[all] oi pessoal!
```

Observações adicionais

Na carga do módulo devem ser informados três parâmetros (número máximo de processos, número máximo de mensagens enfileiradas por processo e tamanho máximo de mensagens). Caso sejam registrados mais processos que o limite, ou ocorra o acúmulo de mensagens (fila circular cheia) ou seja enviada uma mensagem maior que o limite de tamanho, deve ser emitido um alerta no log. No primeiro caso, o processo não será registrado e no último, a mensagem deve ser truncada. Caso mensagens sejam enviadas a processos não registrados, as mesmas deverão ser descartadas pelo driver, e este deverá sinalizar por mensagens de alerta no log. Deve ser possível para um processo efetuar a exclusão do seu registro no driver. Nesse caso, se existirem mensagens enfileiradas para seu nome, as mesmas deverão ser descartadas pelo driver, emitindo-se um alerta no log. O processo de exclusão de registro poderá ser feito por um processo, desde que a solicitação seja realizada pelo mesmo processo que criou o registro (o seu próprio pid).

```
/unreg processo1
```

Para lançar múltiplos processos pelo terminal do Linux embarcado, pode-se carregar a aplicação e em determinado momento colocá-la em *background* usando as teclas Ctrl+Z. Dessa forma, pode-se carregar mais instâncias da mesma aplicação. Para listar os processos colocados em segundo plano, digite *jobs*. Para resumir a execução de um processo, utilize *fg %n* onde *n* representa o número do job.

Apresentação e entrega

Este trabalho deverá ser realizado em duplas ou trios e apresentado no dia 16/05 (apresentação de 5 a 10 minutos). Para a entrega, é esperado que seja enviado pelo Moodle um arquivo *.tar.gz* do projeto com o seu nome e contendo:

1. Código fonte do módulo de kernel desenvolvido;
2. Código fonte da aplicação de usuário desenvolvida para validação do módulo;
3. Scripts para compilação e instalação do módulo e aplicação na distribuição Linux.