Filas de Mensagens

Laboratório de Sistemas Operacionais

Prof. MSc. João Tavares

Introdução

- Três modelos de comunicação
  - Filas de mensagens
  - Semáforos
  - Memória compartilhada

- Duas APIs
  - XSI IPC → tradicional, inspirada na API do System V
  * Utiliza namespace independente e utilitários específicos
  * Amplamente disponível; utilizada em aplicações clássicas como o servidor X11
  - POSIX IPC → sugestão para novas aplicações
  * Utiliza namespace e utilitários do sistema de arquivos
  * Não disponível no Linux antes do kernel 2.6

POSIX Message Queues

- Comunicação orientada a mensagens
- **Mensagens** têm um conteúdo e uma prioridade e são enviadas para, ou recebidas de, uma **Fila**
  - API garante separação entre as mensagens
  - Permite desacoplamento temporal entre processos
  - Tamanho limitado pelo SO (tipicamente 8KB)
- **Fila** → semelhante a uma mailbox
  - Kernel mantém uma lista de mensagens enviadas para a fila, ordenada por prioridade,
    * dentro de uma prioridade, por idade (First-In, First-Out)
  - Possui um nome e permissões como um arquivo
  - Possui uma capacidade limitada

POSIX Message Queues

- **Características gerais das filas**:
  - Nomes globais, descritores locais
  - Persistência de sistema em memória RAM
  - Por padrão, oferece sincronização automática:
  * Envio bloqueia e se fila está cheia
  * Recepção bloqueia se fila está vazia
- **Para se comunicarem, os processos precisam**:
  - Ter credencial compatível com as ACLs (permissões) configuradas para a fila
  - Obter descritor para a fila:
  * A partir do nome da fila; ou
  * Por herança do processo pai
  - Definir um protocolo de acesso e formato de dados

Visão geral da API POSIX MQ

- Operações da API:
  - **mq_open** → cria ou obtém descritor para fila
  - **mq_close** → libera descritor não mais utilizado
  - **mq_getattr / mq_setattr** → consulta/modificação de atributos da fila
  - **mq_notify** → registro de notificação síncrona sobre chegada de mensagens na fila
  - **mq_receive / mq_timedreceive** → recebimento de mensagen da fila
  - **mq_send / mq_timedsend** → envio de mensagens para a fila
  - **mq_unlink** → remove fila do sistema
- Deve-se ligar (*link*) programa com a biblioteca POSIX RT (**-l rt**)

mq_open()

API

```
mqd_t mq_open(*name, oflags);
mqd_t mq_open(*name, oflags, mode, *attrs);
```

- Obtém um descritor para uma fila de mensagens, opcionalmente, criando a fila caso não exista
  - **name** → semelhante a um nome absoluto de arquivo, deve iniciar com '/'. Ex.: “/nomedafila”
  - **oflags** → o que será feito com a fila (O_RDONLY, O_WRONLY, O_RDWR)
    * Opcional: O_CREAT, O_EXCL, O_NONBLOCK
  - **mode** → permissões a serem atribuidas a fila se for criada (O_CREAT)
  - **attrs** → ponteiro para struct definindo tamanho da fila, limite por mensagem, etc (NULL para default)
- **Retorna**: descritor ou -1 em caso de erro

mq_send()

API

```
int mq_send(mqd, *msg_ptr, msg_len, msg_prio);
```

- Adiciona uma mensagem a uma fila, respeitando a ordem de prioriodade
  - **mqd** → descritor da fila destino
  - **msg_ptr** → ponteiro para a mensagem
  - **msg_len** → tamanho da mensagem
  - **msg_prio** → prioridade da mensagem
  * Menor prioridade: 0
  * Maior prioridade: padrão define mínimo 20

- **Retorna**: 0 em caso de sucesso, -1 em caso de erro

mq_receive()

API

```
ssize_t mq_receive(mqd, *buf_ptr, buf_len, *msg_prio);
```

- Recebe próxima mensagem da fila, considerando a prioridade e ordem de chegada das mensagens
  - **mqd** → descritor da fila origem
  - **buf_ptr** → ponteiro para buffer onde mensagem será copiada
  - **buf_len** → tamanho máximo do buffer
  - **msg_prio** → ponteiro para variável inteira que será preenchida com a prioridade da mensagem recebida
- Bloqueia se nenhuma mensagem estiver disponível

- **Retorna**: número de bytes da mensagem recebida, ou -1 em caso de erro

mq_close()

API

```
int mq_close(mqd);
```

- Libera descritor de fila de mensagens
  - **mqd** → descritor da fila

- Raciocínio semelhante a arquivos
  - Limite de número de descritores por processo
  - Herança de descritores em fork()/exec()

- **Retorna**: 0 se sucesso, ou -1 em caso de erro

mq_unlink()

API
```
int mq_unlink(name);
```

- Remove uma fila de mensagens do sistema, descartando conteúdo não consumido
  - **name** → nome da fila a ser removida

- Ocorre em duas fases:
  - Fila é tornada inacessível imediatamente para **mq_open**()
  - Memória só é de fato liberada depois que todos os processos que fecharem os descritores da fila

- **Retorna**: 0 se sucesso, ou -1 em caso de erro

Operações avançadas

- **mq_timedsend**(), **mq_timedreceive**()
  - Variações de mq_send e mq_receive
  - Permitem definir um limite de tempo para o bloqueio das operações de envio e recepção
  - Caso o tempo limite seja atingido, as operações são canceladas e retornam um código de erro para o chamador
- **mq_notify**()
  - Permite a **exatamente 1 processo** monitorar uma fila e receber avisos sobre a chegada de mensagens
    * Através da entrega de um sinal, escolhido pelo processo
    * Através do disparo de uma função (semelhante a uma thread)
  - Ver exemplo na man page da função

Leituras complementares

- STEVENS, W.R. Advanced Programming in the UNIX Environment. 2nd. Ed., Addison Wesley, 2005.

- Man pages
  - Referentes a cada uma das funções abordadas
  - Overview de POSIX Message Queues
    * man 7 mq_overview

- Livro: Advanced Linux Programming
  http://www.advancedlinuxprogramming.com/alp-folder

- Na web: System Software Unix IPC API
  http://jan.newmarch.name/ssw/ipc/unix.html

Referências Bibliográficas

- Material originalmente elaborado por Prof. Cristiano Costa. Material autorizado e cedido pelo autor. Revisado e atualizado por Prof. Luciano Cavalheiro e posteriormente pelo Prof. João Tavares.