Semáforos

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

Semáforo

- Semáforo → contador usado para controlar acesso a dados compartilhados entre múltiplos processos
  - Tipicamente, o valor do contador indica o número de instâncias livres do recurso
- Para usar um recurso compartilhado:
  1. Testar o semáforo que controla o recurso;
  2. Se o valor do semáforo é positivo
     * O processo pode usar o recurso.
     * Processo decrementa o semáforo em 1, indicando que uma instância do recurso foi reservada para seu uso;
  3. Caso contrário, se o semaforo é 0, o processo dorme até o valor ser > 0.
     * Quando acorda retorna ao passo 1.

Semáforo

- Quando o processo termina de usar o recurso
  - O semáforo é incrementado em 1 para indicar que uma instância do recurso foi liberada
  - Se algum processo está dormindo, esperando o semáforo, ele é acordado.

- Para implementar semáforo o teste do valor e o decremento desse valor devem ser uma operação atômica
  - Normalmente implementados pelo kernel

- Caso frequente → semáforos binários (*mutex*)

APIs disponíveis

- POSIX
  - semáforos nomeados
    * implementação sobre sistema de arquivos virtual (VFS)
  - **semáforos anônimos** (*unamed*)
    * implementação em memória
    * para sincronização entre threads
      - estrutura global ou alocada no heap
    * para sincronização entre processos
      - deve ser alocado em mem. compartilhada (ex.: criada com shmget)
- XSI / System V (API legada)
  - arrays de semáforos

Semáforos anônimos POSIX: sem_limit()

API

```
int sem_init(sem_t *sem, int pshared, unsigned int value);
```

- Prepara o semáforo apontado por ***sem*** para o uso
  - **pshared** != 0
    * indica que será utilizado para sincronização entre processos
    * **sem** → deve estar em memória compartilhada
  - **value**
    * contém o valor inicial do contador do semáforo

- Programa deverá ser ligado a biblioteca **pthread** (*-l pthread* no gcc)

Semáforos anônimos POSIX: post e wait

API

```
int sem_post(sem_t *sem);

int sem_wait(sem_t *sem);
```

- Operações clássicas de manipulação do semáforo

- **sem_post**
  - incrementa contador
  - acorda 1 processo ou thread que esteja bloqueado

- **sem_wait**
  - espera até que o contador seja positivo, então decrementa o contador

Semáforos anônimos POSIX: post e wait

API

```
int sem_destroy(sem_t *sem);
```

- Libera recursos alocados para implementação do semáforo criado pelo **sem_init**()

- É boa prática remover estruturas não usadas
  - Evitar “acidentes”
  - Pode haver um limite no número de semáforos que podem ser criados por um processo/thread

- Mais informações: *man sem_overview*

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 Semaphores
    * man 7 sem_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.