MAC 438 - Programação Concorrente - Primeiro Semestre de 2006

Primeiro Exercício-Programa: Sinais e Memória Compartilhada no Unix

Entrega: até 18 de abril de 2006

Este exercício deve ser desenvolvido em equipes de duas pessoas, a fim de suscitar discussão. Ele é pequeno, mas envolve detalhes que devem ser conhecidos por todos vocês. Não deixe seu colega de equipe fazer tudo sozinho!

Dúvidas sobre o enunciado devem ser enviadas para a lista de discussão de MAC-438.

1 O problema

Neste exercício trabalharemos com uma simulação simplória de um sistema de atendimento ao cliente (SAC, ou *call center*). A nossa simulação incluirá apenas os seguintes aspectos, ambos relacionados com o acompanhamento da carga no *call center*: o número de chamadas em atendimento em um dado instante e o histórico diário de atendimento para cada um dos operadores que atendem a chamadas.

Nosso modelo de simulação é composto por dois programas: um programa chamado `operador` e um programa chamado `supervisor`. Esses dois programas interagem com o usuário da simulação através da entrada e da saída padrão.

O programa `operador` modela um atendente do *call center*. Ele lê a entrada padrão e interpreta os seguintes comandos:

- `startcall`
  Indica que o atendente está atendendo a uma nova chamada. Do ponto de vista de nossa simulação é aceitável que um atendente atenda a várias chamadas simultaneamente.
- `endcall`
  Indica que uma das chamadas sendo atendida pelo operador está terminando.
- `report`
  Imprime o número de chamadas sendo atendidas no momento e o número de chamadas já completadas pelo atendente.
- `exit`
  Encerra a execução do programa.

Quaisquer outros comandos devem ser ignorados pelo programa `operador`.

O programa `supervisor` não aceita comandos do usuário via entrada padrão. Ele imprime regularmente as seguintes informações na saída padrão:

1. o número de chamadas em andamento no sistema como um todo (isto é, o total de chamadas sendo atendidas pelos vários operadores ativos) e o número de chamadas já concluídas pelo sistema como um todo;
2. o número de chamadas sendo atendidas e o número de chamadas já concluídas por operador. Em outras palavras: para cada um dos operadores ativos, deve ser impresso o número de chamadas que esse operador está atendendo no momento e o número de chamadas cujo atendimento foi concluído pelo operador.

A impressão dessas informações é feita a cada `n` segundos, onde `n` é um argumento fornecido (na linha de comando) pelo usuário do programa `supervisor`.

Ambos os programas iniciam seu trabalho imprimindo sua identificação, isto é, o número do processo em execução (veja a *man page* para `getpid`). Para executar o programa `operador`, o usuário precisa conhecer a identificação do `supervisor` correspondente. Essa identificação é um argumento fornecido (na linha de comando) pelo usuário do programa programa `operador`.

2 Um “Empurrão Inicial”

Uma “solução incompleta” para este problema está disponível na página do curso. Essa solução, escrita em C e baseada na biblioteca de sinais do Unix, implementa programas `operador` e `supervisor`. Entretanto o programa `supervisor` só imprime as informações referentes ao sistema como um todo, especificadas no ítem 1 da página anterior. Não são impressas as informações por operador, especificadas no ítem 2 da página anterior.

3 Suas Tarefas

- Estude a biblioteca de sinais do Unix e entenda completamente a “solução incompleta”. O material sobre *Unix signals* está disponível na pasta 31 do xerox do CAMAT.
- Pense em como estender a funcionalidade da “solução incompleta” para que ela imprima também as informações por operador. Você notará que a biblioteca de sinais já não será adequada para a comunicação entre os processos `operador` e o processo `supervisor`. Explicar o motivo!
- Escreva uma “solução completa” usando memória compartilhada entre os processos, isto é, com os vários processos `operador` mantendo seus contadores de chamadas numa área que pode ser lida pelo processo `supervisor`. A área de memória compartilhada deve ser implementada através das primitivas de *shared memory* disponíveis no Unix. O material sobre *shared memory* também está na pasta 31 do xerox do CAMAT. Um exemplo de uso (dois processos que compartilham um *buffer* — um escreve e outro lê) está disponível juntamente com a “solução incompleta”.

  Sua solução deve ser escrita em C. Escreva-a de modo que o acesso à memória compartilhada (onde estão os vetores cujos elementos guardam os números de chamadas em atendimento ou já completadas pelos vários operadores, ou alguma outra estrutura de dados que você achar conveniente) seja uma região crítica que é executada com exclusão mútua, através do algoritmo “Tie Breaker” ou do “Bakery Algorithm”. (Escolha um deles e documente bem sua escolha.)

4 Sobre a entrega

Você deverá entregar um arquivo tar.gz contendo os seguintes ítens:

- arquivos-fonte, `makefile`, arquivo `README`, ...
- um relatório sobre sua solução, com respostas para as questões acima.

O desempacotamento do seu arquivo tar.gz deverá produzir um diretório contendo esses ítens. O diretório deve ter nome da forma `ep1-membros-da-equipe` (exemplo: `ep1-joao-maria`).

A entrega será via Internet. Detalhes adicionais serão divulgados na lista de discussão da disciplina.

EPs atrasados não serão aceitos.

Bom trabalho!