Sistemas Operacionais

Bruno Tomé - 0011254[1], Ronan Nunes - 0011219[1]

[1]Instituto Federal de Minas Gerais (IFMG)
São Luiz Gonzaga, s/nº - Formiga / MG - Brasil

ibrunotome@gmail.com, ronannc12@gmail.com

Abstract. *Report about the job of Operating Systems, using the concept of monitors and working with threads.*

Resumo. *Relatório sobre o trabalho de Sistemas Operacionais, usando o conceito de monitores e trabalhando com threads.*

1. Introdução

Este trabalho tem por objetivo fazer com que nós experimentássemos na prática a programação de problemas de sincronização entre processos e os efeitos da programação concorrente, utilizando os recursos de threads POSIX (pthreads) em particular, mutexes (pthread_mutex_t) e variáveis de condição (pthread_cond_t).

2. Resumo do projeto

Nosso programa começa com uma estrutura, criada com objetivo de armazenar a id do personagem e o número de iterações que os personagens terão com a variável compartilhada, o forno. Logo após a estrutura, estão as variáveis globais, elas devem ser globais pois todos os personagens manipularão esses dados em qualquer parte do código. Temos um vetor de 6 posições, que representa uma fila, e um contador do número de vezes que o personagem usou o forno, esta última visa evitar a inanição dos personagens quando o Raj detecta deadlocks.

A primeira função retorna o nome do personagem baseado em seu id, os id's foram posicionados de forma que a namorada do personagem esteja sempre 3 posições a frente do mesmo. Esse posicionamento facilitaria na hora de codificarmos as prioridades se tivéssemos usado o operador de módulo. Porém, não utilizamos este operador, fizemos as decisões de prioridade uma a uma.

Logo após, temos as 4 funções principais do trabalho, onde o personagem usa o forno, passa algum tempo fazendo outras coisas, quer usar o forno ou foi comer. Nestas duas últimas faz-se o uso de mutex, pois é o momento onde fazemos o uso da variável compartilhada forno.

O void *personagens é responsável pelas chamadas das funções que interagem com o forno, dentro de um while, elas serão chamadas até que o número de iterações recebido seja = 0. O void *argumento recebido por parâmetro é um argumento genérico, que pode ser a id do personagem ou o número de iterações.

O void *raj é responsável por liberar deadlocks quando encontrar algum, ele passa de 5 em 5 segundos como especificado e tenta liberar quem menos usou o forno, numa tentativa de previnir inanição.

dispara_threads_personagens é responsável por disparar as 6 threads dos personagens. Logo abaixo o void dispara_thread_raj dispara a thread referente ao Raj.

No main, temos a checagem de passagem de argumentos via console, o número passado via argumento será o número de iterações que os usuários terão com a variável compartilhada forno.

3. Decisões de projeto

Como o trabalho envolve uma fila de pessoas esperando para usar um forno, resolvemos implementar em forma de fila (só que utilizando vetor) a espera dessas pessoas. Essa decisão foi tomada pois no último período fizemos algo parecido, quando tínhamos que controlar o acesso das pessoas aos caixas eletrônicos de um banco usando fila (só que sem o uso de monitores).

Outra decisão tomada foi a definição de prioridades uma a uma. Utilizando o operador de módulo começou a ficar confuso, então tomamos a decisão de entregar algo que funcione e que sabemos explicar como funciona, do que implementar usando módulo e não entender direito como são atribuídas as prioridades.

4. Como executar o programa

Abra o Terminal e digite:

cd <DIRETÓRIO>

make

./main <NÚMERO DE ITERAÇÕES>

No windows, você pode rodar via netbeans ou se tiver um terminal batch basta seguir os comandos acima.

5. Bugs

1. OS X: Após pesquisar em vários sites, não consegui resolver o problema das pthreads no OS X, o programa rodando nele mostra: Pessoa quer comer, pessoa está comendo, pessoa quer comer, pessoa está comendo...

No domingo 01/02 já até havia escrito aqui que entregaríamos o programa com um segmentation fault na última thread, porém ativei as flags -Wall -pedantic -ansi na hora de compilar e descobri que o erro era na inicialização e local de declaração das pthreads, então entregamos o trabalho apenas com esse bug estranho do OS X.