SISTEMAS OPERACIONAIS 2013.1

Prof. Fernando Parente Garcia

Projeto II – Detecção de Deadlock

Objetivo: Detectar deadlocks utilizando o **modelo matricial**.

Descrição: O projeto consiste de duas classes de threads: **processos** e **sistema operacional**. A classe processos poderá ter várias instâncias, que devem simular os processos solicitando, utilizando e liberando recursos do sistema. A classe sistema operacional terá apenas uma instância e ficará responsável por detectar possíveis deadlocks.

Entradas do Sistema:

Antes de iniciar a simulação, o usuário deverá informar todos os tipos de recursos existentes no sistema, e para cada tipo de recurso informado o usuário deverá configurar os seguintes parâmetros:

- Nome do Recurso (Ex: Impressora)
- Identificador do Recurso (Ex: 1)
- Quantidade de instâncias do recurso (Ex: 5)

Obs.: O número máximo de tipos de recursos é 10.

Thread "Sistema Operacional":

Esta thread tem a função verificar periodicamente (a cada intervalo **Δt**) se existe algum deadlock no sistema. Se houver, ela deve informar ao usuário quais processos estão em deadlock. Ao iniciar a execução da simulação, o programa deverá solicitar ao usuário o intervalo **Δt** (em segundos), e em seguida instanciar a thread sistema operacional.

Threads "Processo":

Estas threads deverão solicitar, utilizar e liberar recursos existentes no sistema. Podem existir até 15 processos rodando "simultaneamente".

Criar thread processo:
Durante a criação de cada processo devem ser definidos os seguintes parâmetros:

- **Id** = identificador do processo.
- **ΔTs =** intervalo de tempo de solicitação (em segundos).
  A cada **ΔTs** o processo solicita a utilização de uma instância de um recurso escolhido **aleatoriamente** dentre os tipos de recursos existentes no sistema. Se o recurso estiver disponível, o processo receberá o recurso, irá utilizá-lo e ao final deverá liberá-lo. Caso o recurso solicitado não esteja disponível, o processo deverá dormir e só deverá ser acordado quando o recurso que ele solicitou for liberado por outro processo.
- **ΔTu =** intervalo de tempo de utilização (em segundos).
  Após o processo ter tomado posse de um recurso, ele deverá utilizá-lo durante o intervalo de tempo **ΔTu** e em seguida liberá-lo.

Eliminar thread processo:
Esta opção permite eliminar um processo a partir de um **id**. Deve ser permitido eliminar tanto os processos que estejam "rodando" quanto os que estejam "bloqueados",

Saídas:

A interface deve mostrar, a cada instante o status de cada processo (rodando ou bloqueado), os recursos existentes, os recursos disponíveis, quais recursos estão sendo utilizados por cada um dos processos, quais recursos estão sendo aguardados pelos processos bloqueados e um "log" na tela mostrando todas as operações efetuadas por todos os processos. Este "log" deverá mostrar mensagens do tipo "*o processo X solicitou/está utilizando/liberou o recurso Y*".
A interface também deverá informar se existe algum deadlock e quais processos estão envolvidos no deadlock.

Data de entrega:

- Engenharia de Computação: **18/10/13**
- Engenharia de Telecomunicações: **21/10/13**