Universidade Federal de Mato Grosso do Sul
Faculdade de Computação
Sistemas Operacionais - T02 - Prof. Irineu Sotoma

Descrição do Trabalho T2 - Escalonamento de threads - 31 de maio de 2017

1. O Trabalho T2 visa utilizar sincronização de processos para resolver o seguinte Problema, assumindo que todas as variáveis e parâmetros indicados são inteiros, salvo menção contrária:

   (a) Deve-se desenvolver um sistema para escalonamento de *num* threads que simule uma fila de prontos, a utilização da CPU e um algoritmo de escalonamento por prioridades preemptivo, com utilização de round-robin.

   (b) Cada thread possui uma identificação *id*, com $0 \leq id \leq num - 1$.

   (c) Assume-se que antes do início da execução do sistema, não há thread alguma usando a CPU e a fila de prontos está vazia.

   (d) Cada thread entra na fila de prontos, inicialmente, no tempo $id \times inter$ segundos, com inteiro $inter \geq 0$, em ordem FIFO (*First In - First Out*).

   (e) Assuma que cada instrução de thread gasta *unidade* milissegundos de CPU.

   (f) As decisões de escalonamento podem ocorrer em qualquer instante de tempo que seja múltiplo inteiro de *unidade* milissegundos.

   (g) Cada thread gasta um total de $2 * (id + 1) * unidade$ milissegundos de tempo de processamento.

   (h) Para cada thread há a variável *estado* que indica se ela está na fila de prontos ("pronto"), executando ("executando") ou terminado ("terminado").

   (i) Para cada thread há variável *resta* que indica quanto resta de seu tempo de processamento.

   (j) Cada thread *id* possui uma prioridade inteira *prio*, $0 \leq prio \leq 100$, definida na inicialização do sistema, sendo que um valor maior significa uma prioridade mais alta.

   (k) Threads de mesma prioridade são desempatadas pela ordem FIFO que estão na fila de prontos.

   (l) O sistema possui um time quantum *quantum* em milissegundos, múltiplo de *unidade*, definido na inicialização do sistema.

   (m) Uma vez que a thread de maior prioridade foi escolhida pelo seu escalonador, ela pode usar a CPU por um tempo bem maior que *quantum* milissegundos enquanto ela continuar a ser a thread de maior prioridade após cada *quantum* milissegundos.

   (n) A cada *quantum* milissegundos que uma thread usa a CPU, *prio* é decrementado de uma unidade, até chegar no limite 0.

   (o) Quando uma thread de maior prioridade ou igual prioridade chegar à fila de prontos, ou quando a prioridade da thread atual ficar menor ou igual à das threads já existentes na fila de prontos, a thread atualmente em execução deve ser retirada da CPU ao esgotar seu *quantum* e colocada de volta à fila de prontos.

   (p) Uma thread pode ser retirada da CPU antes de *quantum* milissegundos apenas se $resta < quantum$.

   (q) No início do período de simulação, a primeira thread inicia a execução.

   (r) A simulação termina quando todas as *num* threads já esgotaram totalmente seus tempos de processamento.

   (s) Utilizar a linguagem Java e os recursos de sincronização Java para resolver o Problema com pelo menos uma thread Java para cada thread, que se interagem via primitivas de sincronização para memória compartilhada.

(t) Durante a execução do seu programa, utilize a hora do sistema como estampa de tempo para mostrar modificação ou ocorrência dos seguintes eventos, juntamente com a lista de informações [*id*: *estado*, *prio*, *resta*] de cada thread do sistema:

i. Início da simulação
ii. A thread *id* chegou inicialmente à fila de prontos
iii. A thread *id* está executando
iv. A thread *id* voltou à fila de prontos
v. A thread *id* esgotou seu tempo total de processamento
vi. Término da simulação

2. O Trabalho T1 é composto da implementação em Java com threads e de relatório sobre a solução para o Problema.

3. Nota:T1 é a nota do Trabalho T1. A seguir, na descrição de cada item que compõe Nota:T1, $\{0, a\}$ significa o valor 0 ou o valor a, e $[0 - a]$ significa algum valor real de 0 até a.

4. Nota:T1 = T1:1 × T1:2 × T1:3 × (T1:4 + T1:5 + T1:6 + T1:7), onde:

T1:1) $\{0, 1\}$: Cada grupo, de **1 ou 2** acadêmicos, deverá desenvolver as implementações em **Java** com **threads** em **Linux** sem a geração de erros de compilação e sem geração de exceções durante a execução.

T1:2) $\{0, 1\}$: O relatório completo (no formato PDF) e o código fonte zipado (**.zip**) da solução deverão ser entregues diretamente via "Entrega do Trabalho T2" de "Sistemas Operacionais - T02" em `http://ead.facom.ufms.br`. Um fórum de discussão deste trabalho já se encontra aberto. Você pode entregar o trabalho quantas vezes quiser até às **19 horas** do dia **21 de junho de 2017**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos. Para prevenir imprevistos como falhas de energia, sistema ou internet, recomendamos que a entrega do trabalho seja feita pelo menos um dia antes do prazo.

T1:3) $\{0, 1\}$: Implementação com pelo menos uma thread Java para cada thread do sistema de simulação.

T1:4) $[0 - 2]$: O relatório completo, de 3 a 6 páginas, DEVERÁ conter o nome de todos os integrantes do grupo, Introdução (motivação e objetivos do trabalho), Desenvolvimento (qual foi a ideia da implementação de cada item da especificação, fazendo menção às classes e métodos implementados, e por quê funciona) e Conclusão (qual foi a importância do trabalho na sua formação e quais problemas foram encontrados no seu desenvolvimento).

T1:5) $[0 - 1]$: Implementação que recebe números inteiros positivos, e quando for indicação de tempo será em milissegundos, separados por espaço como parâmetros de entrada, no formato:

num inter unidade quantum (lista de *id prio*).

Um exemplo de valores de parâmetros de entrada é:

4 300 100 500 0 50 1 52 2 54 3 57.

T1:6) $[0 - 2]$: Implementação que gere a saída dos eventos conforme o item 1t.

T1:7) $[0 - 5]$: Implementação que trate corretamente os eventos a partir do início da simulação.

5. **Caso o professor detecte plágio entre trabalhos, no todo ou em parte, os trabalhos envolvidos terão Nota:T1 = 0.**