Polling Server (PS)

Arquivo fonte: ps.c, ps.cc ou ps.cpp

1. Tarefa

Este trabalho consiste na implementação de um simulador para teste do algoritmo de escalonamento *Polling Server* (PS) com Razão Monotônica, aplicado a tarefas periódicas e aperiódicas.

O simulador deverá: ler da entrada padrão um conjunto de valores que correspondem à definição do tempo de simulação, das características do *Polling Server* (computação, período e *deadline*) e de um conjunto de tarefas (número de tarefas periódicas; número de tarefas aperiódicas; para cada tarefa periódica o tempo de computação, período e *deadline*; para cada tarefa aperiódica o tempo de computação e de chegada), conforme definido na Seção 2 (Entrada); e gerar na saída o resultado da simulação, conforme definido na Seção 3 (Saída).

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém 3 números inteiros que correspondem respectivamente ao tempo de simulação (*T*), ao número de tarefas periódicas (*TP*) e ao número de tarefas aperiódicas (*TA*). A segunda linha de um conjunto de teste contém 3 números inteiros que descrevem o *Polling Server*: tempo de computação (C_{PS}), período (P_{PS}) e *deadline* (D_{PS}, que para esta implementação será sempre igual a P_{PS}). A seguir aparecem na entrada as descrições de cada uma das tarefas: inicialmente a descrição das *TP* tarefas periódicas, seguidas da descrição das *TA* tarefas aperiódicas. A descrição de cada tarefa periódica é composta por três valores que correspondem respectivamente a: tempo de computação da tarefa (C_i), período da tarefa (P_i) e *deadline* da tarefa (D_i). A descrição de cada tarefa aperiódica é composta por dois valores que correspondem respectivamente a: tempo de computação da tarefa (C_i) e tempo de chegada da tarefa (A_i).

Prioridades para as tarefas periódicas serão definidas pelo algoritmo Razão Monotônica, enquanto a ordem de execução das tarefas aperiódicas será definida pela ordem de chegada.

O final das entradas é indicado por $T = 0$, $TP = 0$ e $TA = 0$.

Os valores de entrada devem ser lidos da entrada padrão (normalmente o teclado) – por exemplo, com *scanf()* ou *getchar()*, em C –, de forma que seja possível redirecionar um arquivo para o processo. Não se deve utilizar arquivos de entrada, nem funções para esperar pelo pressionamento de teclas (comuns quando se depura um programa).

Exemplo de Entrada

```
20 2 2
1 5 5
4 10 10
8 20 20
5 1
12 1

20 2 3
1 4 4
2 5 5
2 10 10
2 2
8 1
15 1
```

```
0 0 0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve executar a simulação da execução das tarefas usando o algoritmo PS com Razão Monotônica e mostrar um conjunto de informações sobre esta execução.

A primeira informação a ser mostrada sobre a simulação corresponde a uma simplificação do diagrama de Gantt (que mostra a execução dos processos), representando cada unidade de execução com um caractere associado ao processo que está ocupando o processador neste respectivo tempo. A primeira tarefa descrita na entrada será representada pelo caractere 'A', a segunda tarefa será representada pelo caractere 'B', e assim sucessivamente. Unidades de execução ociosas são indicadas pelo caractere '.' (ponto). E unidades de execução das tarefas periódicas executadas fora do prazo (após o fim de seu *deadline*) são representadas em letras minúsculas.

A linha seguinte deverá mostrar o número de preempções e o número de trocas de contexto. O número de trocas de contexto deverá ser contabilizado considerando-se que cada computação de uma tarefa periódica é executada por uma nova instância de processo (na prática, isto significa que se a computação de um período ocorrer logo após a computação do período seguinte, isto será contabilizado como uma troca de contexto) e que, nos períodos ociosos (representados pelo caractere '.'), o processador executa um processo *idle* (o que implica que há trocas de contexto associadas a este processo). **Importante:** os números de preempções e de trocas de contexto deverão ser contabilizados até o tempo final da simulação, considerando-se todos os eventos que ocorrem até o último tempo de simulação (inclusive).

Como a entrada pode ser composta por vários conjuntos de teste, os resultados de cada conjunto deverão ser separados por uma linha em branco. A grafia mostrada no Exemplo de Saída, a seguir, que corresponde ao resultado esperado para o exemplo de entrada apresentado anteriormente, deve ser seguida rigorosamente.

Os valores de saída devem ser escritos na saída padrão (normalmente o vídeo) – por exemplo, com *printf()* ou *putchar()*, em C –, de forma que seja possível redirecionar a saída gerada pelo processo para um arquivo texto qualquer. Não se deve utilizar arquivos de saída, nem funções para limpar a tela.

Exemplo de Saída

```
AAAABCBBBBAAAABDBB..
4 9

AABBCAA.C.AADBBAEA..
5 14
```

(esta saída corresponde ao exemplo de entrada acima)

4. Caso especias

Na implementação do *Polling Server* com Razão Monotônica, é preciso levar em consideração diferentes situações que podem ocorrer em relação à prioridade dos processos. O *Polling Server* terá sua prioridade atribuída de acordo com o seu período, assim como acontece com os demais processos periódicos. Desta forma, a implementação deverá levar em consideração situações em que o *Polling Server* tenha a menor prioridade, prioridades intermediárias ou a maior prioridade.

Também é preciso levar em consideração a seguinte situação: como, quando não há tarefas aperiódicas para executar no início de seu período, o *Polling Server* abre mão de sua computação, caso ocorram ciclos ociosos, estes ciclos ociosos deverão ser usados para executar tarefas

aperiódicas. Isto também ocorrerá nas situações em que o *Polling Server* já tiver, dentro de seu período, consumido suas unidades de computação: havendo ciclos ociosos, eles serão usados para executar tarefas aperiódicas.

O seguinte exemplo de entrada ilustra esta situação:

```
12 1 1
1 3 3
2 4 4
1 5

0 0 0
```

Para o exemplo acima será produzido o seguinte resultado:

```
AABBAABBABA.
4 8
```

5. Restrições

1 <= T <= 100000 (N = 0 apenas para indicar o final da entrada)

1 <= TP <= 13 (TP = 0 apenas para indicar o final da entrada)

1 <= TA <= 13 (TA = 0 apenas para indicar o final da entrada)

1 <= C_i <= 100000

1 <= P_i <= 100000

1 <= D_i <= 100000

1 <= A_i <= 100000

Autor: Roland Teodorowitsch