# *Earliest Deadline First* (EDF)

*Arquivo fonte: edf.c, edf.cc ou edf.cpp*

## 1. Tarefa

Este trabalho consiste na implementação de um simulador para teste do algoritmo de escalonamento *Earliest Deadline First* (EDF), aplicado a tarefas periódicas, bem como a execução do teste de escalonabilidade baseado na taxa de utilização do processador.

O simulador deverá: ler da entrada padrão um conjunto de valores que correspondem à definição de um conjunto de tarefas (número de tarefas e para cada tarefa o tempo de computação, período e *deadline*), conforme definido na Seção 2 (Entrada); e gerar na saída o resultado da simulação e os valores correspondentes ao teste de escalonabilidade para cada conjunto de tarefas, 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 um número inteiro *N*, que indica o número de tarefas, e um número inteiro *T*, que indica o tempo de simulação. A seguir aparecem na entrada as descrições de cada uma das *N* tarefas. A descrição de cada tarefa 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 ($D_i$). O final das entradas é indicado por $N = 0$ ou $T = 0$.

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**

```
2 20
2 4 4
5 10 10

3 12
2 4 4
1 6 6
3 12 12

0 0
```

## 3. Saída

Para cada conjunto de teste da entrada seu programa deve executar a simulação da execução das tarefas usando o algoritmo EDF e também aplicar o teste de escalonabilidade baseado na taxa de utilização do processador.

A simulação deve ser apresentada em duas linhas: a primeira mostrando uma simplificação do diagrama de Gantt correspondente à execução dos processos e a segunda linha mostrando o número de trocas de contexto e o número de preempções. Na primeira linha, usam-se caracteres para representar unidades de execução no processador. Uma unidade de execução corresponde à execução da primeira tarefa é indicada pelo caractere 'A'. Uma unidade de execução da segunda tarefa, pelo caractere 'B'. E assim sucessivamente. Unidades de execução ociosas são indicadas pelo caractere '.' (ponto). Ciclos de execução das tarefas que forem executados em atraso (após o respectivo *deadline*) deverão ser exibidos em minúsculas.

Para contabilização do número de trocas de contexto e preempções, deve-se realizar a contagem até o tempo de simulação fornecido na entrada, considerando todos os eventos que ocorrerem neste tempo. Também deve-se considerar que as unidades de execução ociosas (indicadas por '.') são executadas por um processo *idle* (ocioso), que também sofre trocas de contexto e preempções (que devem ser contabilizadas). E deve-se considerar ainda que cada ciclo de execução de uma tarefa é executado por uma instância de processo (ou *thread*), o que significa que, entre dois ciclos de execução de períodos diferentes que ocorrem em tempos adjacentes, também ocorrerá uma troca de contexto.

Depois das duas linhas iniciais, apresentam-se o resultado do teste de escalonabilidade baseado na taxa de utilização do processador, calculado a partir da seguinte fórmula (considerando-se a execução em um único processador):

$$U = \sum_i^n U_i = \sum_i^n C_i / P_i \leq 1$$

onde: $C_x$ é o tempo de computação da tarefa *x* e $P_x$ é o período da tarefa *x*.

Para cada conjunto de tarefas apresenta-se o valor de *U* com 4 casas decimais (e arredondamento para o valor mais próximo), seguido de um texto que poderá ser: “OK” (caso $U \leq 1$ ) ou “NOK” (caso $U > 1$ ).

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**

```
AABBAABBBAABAABBAABB
10 3
1.0000 OK

AABCAABCAAC.
9 3
0.9167 OK
```

(esta saída corresponde ao exemplo de entrada acima)

## 4. Restrições

1 <= *N* <= 26 (*N* = 0 apenas para indicar o final da entrada)

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

1 <= $C_i$ <= 100000

1 <= $P_i$ <= 100000

1 <= $D_i$ <= 100000

*Autor: Roland Teodorowitsch*