Estrutura de Dados e Algoritmos
Projeto 03: Escalonador de Processos
Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

1 Contextualização

A CPU é um recurso muito valioso em qualquer computador, e os processos competem pela sua utilização. Para gerenciar o uso da CPU, o sistema operacional conta com o escalonador de processos, que determina a ordem na qual cada processo utilizará a CPU. O seu objetivo é garantir que todos os processos eventualmente sejam atendidos (evitando o *starvation*).

Uma técnica utilizada nos escalonadores de processos é o round-robin. Esta técnica estabelece um *quantum*, pequena fatia de tempo que cada processo utilizará a CPU. Após o término do *quantum*, o escalonador retira o processo que utilizava a CPU e o coloca atrás de todos os outros que aguardam a utilização do recurso.

Neste projeto um escalonador de processos que utiliza a técnica round-robin deverá ser simulado.

2 Especificação

O projeto deverá ser executado através da linguagem `C`.

A entrada deve ser lida da entrada padrão (`stdin`), enquanto a saída deverá ser impressa na saída padrão (`stdout`).

O programa deverá obedecer rigorosamente o formato de saída especificada, pois parte da correção será automatizada.

2.1 Entrada

A primeira linha da entrada contém dois inteiros separados por um espaço: n ($1 \leq n \leq 1000$), indicando o número de processos e q ($1 \leq q \leq 100$), o valor do *quantum* em milissegundos.

As próximas n linhas descrevem cada processo, determinado pela tripla *name*, *pid* e *time*:

- *name*: representa o nome do processo (máximo de 30 caracteres minúsculos).
- *pid* ($1 \leq pid \leq 10^5$): refere-se a um identificador inteiro único para cada processo.
- *time* ($1 \leq time \leq 1000$): refere-se ao tempo de CPU, em milissegundos, requerido pelo processo para que ele finalize a sua execução.

Os valores de *name*, *pid* e *time* estão separados por um espaço.

2.2 Saída

Para cada processo, o programa deverá imprimir uma linha no formato "`<pid> <nome> <final_time> ms`" em ordem de finalização dos processos, em que: `<nome>` corresponde ao nome do processo, `<pid>` ao pid do processo e `<final_time>` ao tempo total que o processo levou até sua finalização.

3 Exemplos

- Entrada

```
5 50
firefox 1000 1000
inkscape 1001 750
terminal 1002 225
inkscape 1003 975
chromium 1004 950
```

- Saída:

```
terminal 1002 1125 ms
inkscape 1001 3125 ms
chromium 1004 3825 ms
firefox 1000 3875 ms
inkscape 1003 3900 ms
```

3.1 Limites de Tempo e Memória

Para cada caso de teste, será permitido a execução do programa por apenas 1 segundo com utilização máxima de 256 MB de memória. Caso o programa leve mais tempo ou memória do que isso, considerar-se-á que o algoritmo empregado foi ineficiente.

3.2 Documentação

Junto do(s) código(s) necessário(s) para resolver o problema, deverá ser disponibilizado um arquivo README, identificando o autor do trabalho e especificando as instruções para compilação e execução do(s) código(s).

3.3 Critérios de Correção

Fazem partes dos critérios de correção:

- Eficiência do programa.
- Utilização de estruturas de dados adequadas.
- Documentação: além do arquivo README, o código deve estar bem documentado.
- Legibilidade.

3.4 Ambiente de Correção

Os projetos serão corrigidos em uma máquina com sistema `GNU/Linux` e compilador `gcc 10.2.0`.

Trabalhos que não compilarem não serão avaliados.

4 Considerações

- Este projeto deve ser executado individualmente.
- Os trabalhos que incidirem plágio serão avaliados automaticamente com nota 0 para os envolvidos. Medidas disciplinares também serão tomadas.
- O trabalho deve ser entregue dentro de uma pasta zipada com a devida identificação do(s) aluno(s) através da sala de aula virtual da disciplina na data estipulada no ambiente.