MC202GH - ESTRUTURAS DE DADOS

2º Semestre de 2017

Professor: Rafael C. S. Schouery

Monitores: Yulle Glebbyo Felipe Borges (PED)

Caio Vinícius Piologo Véras Fernandes (PAD)

Lab. 04 - Escalonamento de Processos

Peso 2

1. O Problema

Em computadores modernos, temos várias aplicações rodando ao mesmo tempo. Com uma unidade processadora limitada, é necessário que o processador faça um escalonamento das tarefas a serem executados em cada ciclo de processamento. Cada aplicação gera diversos processos que são selecionados para serem executados de acordo com suas prioridades para que todos as aplicações funcionem ao mesmo tempo. Os ciclos de processamentos são muito rápidos, o que permite a execução dos processos em partes, possibilitando que vários processos possam ser parcialmente processados em um curto intervalo de tempo.

Uma das políticas de escalonamento de processos empregadas em sistemas atuais é a de *Round Robin*. Neste método, os processos são organizados em uma lista circular de acordo com suas prioridades e, então, o escalonador seleciona o processo que está na primeira posição da lista para ser executado por um ciclo. Caso este processo não seja terminado após este ciclo, o escalonador mantém o processo na lista e segue para executar o próximo.

Nesta tarefa, utilizaremos uma **lista circular duplamente ligada** para implementar a política de escalonamento de processos por Round Robin. Cada processo possui um identificador único e uma duração em número de ciclos de execução, e deve ser inserido em uma determinada posição da lista de execução. O escalonador pode escolher executar os processos por um número limitado de ciclos, o que significa que os processos não necessariamente serão executados até o fim.

2. Entrada

O programa deve permitir operações de inserção e remoção de processos na lista, impressão dos identificadores de todos os processos da lista, priorização de um determinado processo e, por fim, execução dos processos por um determinado número de ciclos. A execução dos processos deve ser feita de acordo com a ordem dos processos na lista no momento da solicitação da execução. Mais informações sobre as operações e suas entradas são encontradas na tabela abaixo.

ID	Nome	Descrição
1	Inserção de um processo	Dado um **inteiro i** representando o identificador do processo, e um **inteiro d** indicando a duração do processo em número de ciclos, e um inteiro **p** representando a posição do processo na fila de prioridades, insere o processo na lista. Posições válidas variam entre 0 e a quantidade de processos já inseridos na lista, sendo que a posição 0 indica o primeiro processo a ser executado.
2	Remoção de um processo	Dado um **inteiro i** representando o identificador do processo já requisitado e remove o processo da lista. Assume-se que existe exatamente um processo com identificador **i** na lista. Caso o primeiro processo da lista seja removido, seu sucessor (próximo) passa a ser o primeiro processo da lista
3	Impressão da lista de processos	Imprime o identificador e a duração de cada processo na lista, respeitando a ordem da lista. Cada processo é representado pela tupla (identificador,duração), e são separados um do outro por espaços em branco
4	Priorizar um processo	Dado um **inteiro i** representando o identificador de um processo já requisitado, passa este processo para o início da fila de execução, mantendo o restante da fila sem alterações. Assume-se que existe exatamente um processo com identificador **i** na lista
5	Executar Processos	Dado um inteiro **q**, executa os próximos **q** ciclos de processamento, onde cada processo é executado por um ciclo. Após executar um processo, é necessário decrementar sua duração, e caso a duração de um processo atinja valor 0 após a execução de um ciclo, este processo deve sair da lista. Ao fim de cada ciclo, a lista segue para o próximo processo, e como

		estamos utilizando uma lista circular, o processo que acabou de ser executado passa a ser o último da lista (caso sua duração ainda seja diferente de 0)
0	Finalização	Saia do sistema, liberando corretamente toda a memória utilizada

Exemplo de entrada:

```
1
101 3 0
1
104 2 1
1
103 2 2
1
102 4 3
5 5
2 102
4 101
3
5 3
0
```

3. Saída

As saídas de cada operação são descritas em detalhe na tabela abaixo.

ID	Saída	Comentário
1	Processo **X** requisitado com duração de **Y** ciclo(s)!	**X** é o identificador do processo requisitado e **Y** é sua duração em ciclos
2	Processo **X** removido!	**X** é o identificador do processo a ser removido
3	(X1,D1) (X2,D2) (X3,D3) … (Xn,Dn)	**X1**, **X2**, **X3** e **Xn** são identificadores dos processos de acordo com a ordem da fila atua, e **D1**, **D2**, **D3** e **Dn** são as durações destes processos respectivamente. Obs: Deve existir um caractere de espaço branco depois do último identificador, antes do `\n`

4	Priorizando o processo **X**!	**X** é o identificador do processo a ser priorizado
5	(X1,D1) (X2,D2) … (Xn,Dn) *ou* Todos os processos terminaram!	**X1**, **X2**, e **Xn** são os identificadores dos processos que sobraram após a execução de q ciclos de processamento. **D1**, **D2** e **Dn** são as durações resultantes dos respectivos processos sobreviventes. Neste caso, o processo **Xn** foi o último (que não terminou) a ser executado neste comando, e o processo **X1** será o próximo a ser executado caso haja uma outra requisição. Caso não todos os processos sejam removidos da lista antes da execução de **q** ciclos, deve-se imprimir a frase "Todos os processos terminaram!" e finalizar o comando. Obs: Na impressão da lista de processos sobreviventes, deve haver um caractere de espaço branco após a tupla representando o último processo, antes do `\n`.
0	Sistema encerrado.	Antes de encerrar o sistema, é necessário liberar toda a memória alocada

Exemplo de saída:

```
Processo 101 requisitado com duração de 3 ciclo(s)!
Processo 104 requisitado com duração de 2 ciclo(s)!
Processo 103 requisitado com duração de 2 ciclo(s)!
Processo 102 requisitado com duração de 4 ciclo(s)!
(104,1) (103,1) (102,3) (101,1)
Processo 102 removido!
Priorizando o processo 101!
(101,1) (104,1) (103,1)
Todos os processos terminaram!
Sistema encerrado.
```

4. Informações

- Este laboratório possui **peso 2**.
- **Não** há um número máximo de submissões.

- No início de cada arquivo a ser submetido, insira um comentário com seu nome, RA e uma breve descrição do conteúdo do arquivo.
- Apenas comentários no formato `/* comentário */` serão aceitos. Comentários com `//` serão acusados como erros pelo SuSy.
- Os cabeçalhos permitidos são:
    - `stdio.h`
    - `stdlib.h`
    - `math.h`
- A submissão da sua solução deverá conter múltiplos arquivos:
    - **lista.h**: interface da lista circular duplamente ligada
    - **lista.c**: implementação da interface da lista circular duplamente ligada
    - **lab04.c**: cliente que utiliza a estrutura
- Será disponibilizado um **Makefile** para este trabalho na página do laboratório:
    - Para compilar seu projeto, basta navegar até o diretório do projeto e utilizar o comando ‘make’ no terminal do Linux.
    - Veja mais instruções na página da disciplina.

5. Critérios de Avaliação

- Seu código deverá passar por todos os casos de teste do SuSy definidos para este laboratório. Caso positivo, utilizaremos os seguintes critérios adicionais:
    - Indentação correta de todo o código. Escolha entre espaços ou tabs para identar seu código e seja coerente em todos os arquivos. Indentações utilizando espaços e tabs intercalados, ou trechos não indentados corretamente irão resultar em uma penalização de **um ponto** na nota
    - A **não utilização da estrutura de lista circular duplamente ligada** para a representação da lista de processos neste laboratório implicará em nota **zero**
    - Utilização de bibliotecas ou funções não permitidas pelo enunciado resultará em uma **penalização severa** na nota
    - Toda a memória alocada deve ser liberada adequadamente ao fim do programa