Trabalho de Linguagem de Programação – LPRO Ref. 2° Bimestre
1° Semestre de 2006
Professor Marcelo Santos

Parte 1: Instruções

1. Este trabalho corresponde à avaliação referente às aulas práticas da disciplina (50% da nota de 2º bimestre);
2. **Prazo para Entrega: até 03/06/2006 (sem prorrogação do prazo), no início da prova (P2);**
3. O trabalho poderá ser feito em grupo de até **CINCO** alunos;
4. Material para entregar: LISTAGEM DO CÓDIGO FONTE + DISQUETE CONTENDO O CÓDIGO FONTE (*.C OU *.CPP) – Coloque o material acima (listagem + disquete) num envelope de papel ou plástico, identificado com o nome dos componentes do grupo;
5. Os programas serão corrigidos utilizando o compilador Turbo C (versão disponível no laboratório);
6. Devem ser utilizados módulos (funções e procedimentos);
7. Entrada (além do arquivo de dados) e saída de dados deverão ser feitas em modo DOS (prompt de comando – tela de texto);
8. **Critérios de avaliação:**
   a) **Funcionamento, clareza na elaboração do algoritmo (codificação), uso adequado de variáveis (atenção quanto ao uso desncessário), documentação (comentários ao longo do programa), desenho interface com o usuário;**
   b) **Entrevista (questões sobre a implementação do trabalho, que deverão ser respondidas juntamente com a avaliação escrita – sem consultas ao trabalho)**

   Composição da Nota (Aula Prática) = item_a*0.5+item_b*0.5

   Observação: Para que seja considerada a nota da entrevista, deve ser entregue o material (item 4).
9. Trabalhos iguais ou muito parecidos com os de outros grupos/alunos serão **ANULADOS** (Nota ZERO na avaliação prática, inclusive a entrevista)
10. **Não serão aceitos trabalhos enviados por e-mail.**

Parte 2: O trabalho

Escalonamento de Processos num Sistema Operacional

O escalonamento de processos refere-se como os processos são distribuídos para execução nos processadores num Sistema Computacional. De acordo com Tanenbaum:
"Quando mais de um processo é executável, o Sistema Operacional deve decidir qual será executado primeiro. A parte do Sistema Operacional dedicada a esta decisão é chamada escalonador (*scheduler*) e o algoritmo utilizado é chamado algoritmo de escalonamento (*scheduling algorithm*)."
A forma com que ocorre o escalonamento é, em grande parte, responsável pela produtividade e eficiência atingidas por um Sistema Computacional. Mais do que um simples mecanismo, o escalonamento deve representar uma política de tratamento dos processos que permita obter os melhores resultados possíveis em um sistema.

A) Escalonamento Preemptivo versus Não Preemptivo

Um algoritmo de escalonamento é dito **não preemptivo** quando o processador designado para um certo processo não pode ser retirado deste até que o processo seja finalizado. Por outro lado, um algoritmo de escalonamento é considerado **preemptivo** quando a CPU designada para um processo pode ser retirada deste em favor de um outro processo. Algoritmos preemptivos são mais adequados para sistemas onde múltiplos processos requerem a atenção do sistema, ou seja, no caso de sistemas multiusuário interativos (sistemas de tempo compartilhado) ou em sistema de tempo real. Nestes casos, a preempção representa a mudança do processo em execução. Deste modo, para que a CPU seja retirada de um processo, interrompendo a execução deste, e designada a outro processo, anteriormente interrompido, é fundamental que ocorra a mudança de contexto dos processos. Tal mudança exige que todo o estado de execução de um processo seja adequadamente armazenado para sua posterior recuperação, representando uma sobrecarga computacional para a realização desta mudança e armazenamento de tais dados. Usualmente, os algoritmos preemptivos são mais complexos devido à natureza imprevisível dos processos.
Por sua vez, os algoritmos não preemptivos são mais simples, se comparados aos preemptivos, e adequados para o processamento não interativo, como no caso do Processamento em Lote. Embora não proporcionem interatividade, são geralmente mais eficientes e previsíveis quanto ao tempo de entrega de suas tarefas.

B) Escalonamento *Round Robin*

O escalonamento *Round Robin*, ou cirular, é um dos mais utilizado. Nesse escalonamento, os processos também são organizados numa fila segundo a ordem de chegada e então são despachados para execução pelo processsador. Entretanto, ao invés de serem executados até o final, a cada processo é concedido apenas um pequeno intervalo de tempo para uso do processador, denominado fatia de tempo, *time-slice*. Caso o processo não seja finalizado neste intervalo de tempo, ocorre sua substituição pelo próximo processo na fila de **processos prontos**, sendo o processo interrompido colocado no **final da fila de pronto**. Isto significa que, ao final da fatia

de tempo do processo, ocorre a preempção do processador, ou seja, o processador é designado a outro processo, sendo salvo o contexto do processo interrompido para permitir a continuidade da execução deste processo quando chegar o seu momento novamente.
Por outro lado, caso o processo que está em execução necessite fazer uma operação de entrada e/ou saída, este vai para a fila de processos aguardando o recurso de entrada e saída, ou diretamente para o uso do recurso de entrada/saída, caso não tenha nenhum processo executando a entrada e saída.
O escalonamento *Round Robin* se baseia na utilização de temporizadores, sendo um algoritmo preemptivo e bastante adequado para ambientes interativos, ou seja, em sistemas de tempo compartilhado onde existem múltiplos usuários simultâneos sendo, portanto, necessário garantir tempos de resposta razoáveis. A figura (Figura 1) abaixo mostra resumidamente o esquema do escalonamento *Round Robin*.

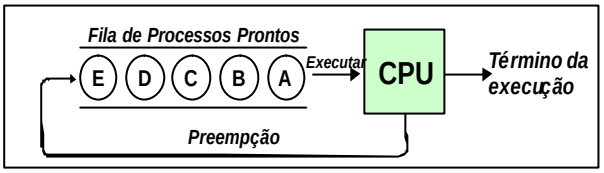

Figura 1. Resumo do escalonamento *Round Robin.*

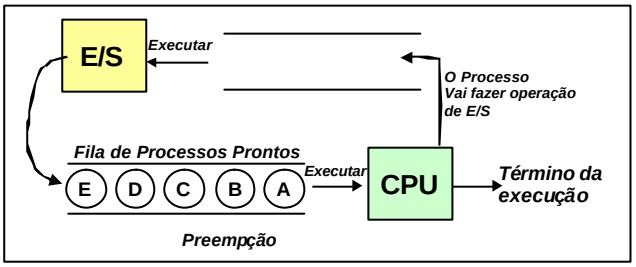

Figura 2. Fluxo do escalonamento *Round Robin*, incluindo a operação de entrada e/ou saída (E/S).

C) Trabalho

Escreva um programa em C que implemente o algoritmo de *Round Robin* para fazer o escalonamento de um conjunto de processos, de modo que seja mostrada a tabela de execução, conforme apresentada no exemplo descrito no item D. Requisitos:

- O programa deverá utilizar lista encadeada (criadas por meio de alocação dinâmica) para organizar as filas de processos (prontos e aguardando E/S), funções ou procedimentos, estruturas (registros);
- Não é permitido o uso de variáveis globais, exceto constantes;
- O programa deverá estar documentado (com comentários ao longo do mesmo para posterior entendimento);
- O usuário fornece o instante de tempo final da tabela de execução (vide exemplo no item D), onde cada linha deverá ser incrementada de um em um até o instante final (em milésimos de segundo);

- O tempo destinado pelo processador (CPU) para a execução de cada processo é **3ms** (três milésimos de segundo);
- O recurso de entrada e saída implementa um algoritmo de escalonamento, não-preemptivo, baseado em fila.
- Cada processo terá, inicialmente, o seguinte conjunto de informações da tabela abaixo:

Campo	***Tipo***	***Descrição***
Nome	caracter (um caractere)	Informação que identifica o processo (nome do processo, propriamente dito)
Chegada	Inteiro	Instante em que o processo foi criado (em milésimos de segundo). *** Suponha que todos os processos tenham chegado no instante 0 (zero) - o que vale é a seqüência das linhas contidas no arquivo ***
Tempo_CPU	Inteiro	Tempo de uso da CPU, exigido pelo processo (em milésimos de segundo). Este tempo é o tempo de execução, sem contar o tempo de entrada e saída.
Intervalo_E/S	Inteiro	Após o processo ter usado a CPU por um determinado tempo, em função desse intervalo, o processo deverá sair da CPU e fazer uma operação de E/S. (em milésimos de segundo).
Tempo_E/S	Inteiro	Tempo gasto nas operações de E/S – Quantidade de tempo em que o processo fica utilizando o recurso de E/S. (em milésimos de segundo).

- A entrada das informações (características) de cada processo será por meio de um arquivo texto, contendo a seguinte estrurtura (somente os valores de cada campo, separados por espaço em branco, de acordo com a especificação dos campos acima e tipos apresentados acima):

```
<Nome> <Chegada> <Tempo_CPU> <Intervalo_E/S> <Tempo_E/S>
```

- A saída do será a seguinte tabela (veja o exemplo a seguir):

Instante	*Fila de processos prontos*	*Processo em execução (CPU)*	*Fila de Processos Aguardando Recurso de E/S*	*Processo fazendo Operação de E/S*	*Fila de Processos Finalizados*
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

- Se a fila a fila de processos prontos estiver vazia e a CPU ociosa, o processo vai direto para a CPU;

- Se a fila a fila de processos aguardando recurso de E/S estiver vazia e o recurso de E/S ocioso, o processo vai direto para o recurso de E/S;
- Sempre que um processo sai do recurso de E/S, este volta para a fila de processos prontos, mesmo que já não tenha mais necessidade de realizar o processamento. Só depois dessa fila, após passar pela CPU, é que irá para a fila de processos prontos;
- Se houver dois processos para entrar na fila de processos prontos, terá prioridade o processo que estiver saindo da CPU, seguido do processo que estiver saindo do recurso de E/S e, por fim, um novo processo a ser inserido;

C) Exemplo

- O arquivo a seguir contém 4 processos (A, B, C, D) que serão executados pelo programa desse trabalho. Conteúdo do arquivo de entrada:

```
A 0 2 1 2
B 0 6 0 0
C 0 5 2 3
D 0 7 2 1
```

- Significado das linhas do arqquivo:

A -	demora 2ms para ser executado (sem contar o tempo de espera de Entrada e Saída). A cada 1 ms de execução é feita uma operação de Entrada e Saída (E/S) que demora 2 ms para ser concluída.
B -	demora 6 ms para ser executado e não faz operações de E/S.
C -	demora 5 ms para ser executado e (sem contar o tempo de espera por E/S). Depois de ser executado por 2 ms ele faz uma operação de E/S que demora 3 ms para ser concluída.
D -	demora 7 ms para ser executado (sem contar o tempo de espera de Entrada e Saída). A cada 2 ms de execução é feita uma operação de Entrada e Saída (E/S) que demora 1 ms para ser concluída

- Saída:

Legenda:

PROCESSO[<tempo restante de CPU>;<tempo restante de E/S>]

Instante	*Fila de processos prontos*	*Processo em execução (CPU)*	*Fila de Processos Aguardando Recurso de E/S*	*Processo fazendo Operação de E/S*	*Fila de processos finalizados*
0	D[7; 0]-C[5; 0]-B[6; 0]-A[2;0]				
1	D[7; 0]-C[5; 0]-B[6; 0]	A[2; 0]			
2	D[7; 0]-C[5; 0]	B[6; 0]		A[1; 2]	
3	D[7; 0]-C[5; 0]	B[5; 0]		A[1; 1]	
4	A[1; 0]-D[7; 0]-C[5; 0]	B[4; 0]			
5	B[3; 0]-A[1; 0]-D[7; 0]	C[5; 0]			
6	B[3; 0]-A[1; 0]-D[7; 0]	C[4; 0]			

7	B[3; 0]-A[1; 0]	D[7; 0]		C[3; 3]	
8	B[3; 0]-A[1; 0]	D[6; 0]		C[3; 2]	
9	B[3;0]	A[1;0]	D[5;1]	C[3; 1]	
10	C[3;0]	B[3;0]		D[5;1]	A
11	D[5;0]-C[3;0]	B[2;0]			A
12	D[5;0]-C[3;0]	B[1;0]			A
13	D[5;0]	C[3;0]			A-B
14	D[5;0]	C[2;0]			A-B
15		D[5;0]		C[1;3]	A-B
16		D[4;0]		C[1;2]	A-B
17			D[3;1]	C[1;1]	A-B
18		C[1;0]		D[3;1]	A-B
19		D[3;0]			A-B-C
20		D[2;0]			A-B-C
21		D[1;0]			A-B-C
22					A-B-C-D

Bom Trabalho.