Trabalho Prático: Implementação de um Simulador de Escalonamento Round Robin

Disciplina: Arquitetura de Sistemas Operacionais.

Objetivo: Implementar um simulador de escalonamento de processos usando o algoritmo **Round Robin (RR)**. Os alunos receberão os arquivos de cabeçalho (.h) básicos e deverão desenvolver toda a lógica de implementação nos arquivos .c, seguindo rigorosamente as especificações fornecidas.

1. Contexto

Você deve construir um simulador de processos para um sistema operacional hipotético que utiliza o algoritmo Round Robin para escalonamento. O simulador deve:

- Criar processos dinamicamente.
- Gerenciar uma fila circular de processos prontos.
- Alternar a execução entre processos com base em um quantum fixo.
- Lidar com estados de processos (Pronto, Executando, Bloqueado, Terminado).

2. Arquivos Fornecidos

Os alunos deverão fazer download dos arquivos de cabeçalho (.h) que está disponível no Moodle.

3. Requisitos de Implementação

Os alunos devem implementar todos os arquivos .c correspondentes aos *headers* fornecidos, além de um programa principal (main.c).

A. process.c

- Implementar as funções:
  - **create_process**: Aloca e inicializa um processo.
  - **destroy_process:** Libera a memória do processo.
  - **print_process:** Exibe informações do processo no formato:

```
PID: 1 | Estado: PRONTO | CPU: 0 | Restante: 5 | Quantum: 4
```

B. scheduler.c

- Implementar as funções:
    - **init_scheduler**: Inicializa o escalonador com o quantum especificado.
    - **add_to_ready_queue:** Adiciona um processo à fila circular de prontos.
    - **schedule:** Escalona o próximo processo, decrementando seu quantum e tempo restante.
    - **print_queue:** Exibe todos os processos na fila de prontos.

C. rr_scheduler.c

- Implementar funções auxiliares para:
    - **handle_blocked_process**: Remove um processo bloqueado da fila de prontos.
    - **handle_terminated_process:** Remove um processo terminado e libera sua memória.

D. main.c

- Simular a criação de processos com tempo de execução aleatório.
- Gerar eventos aleatórios (bloqueio/desbloqueio).
- Chamar as funções de escalonamento e exibir o estado da fila a cada ciclo.

4. Funcionalidades Obrigatórias

- Fila Circular de Prontos:
    - Processos em estado PRONTO devem ser organizados em uma fila circular.
    - O processo em execução (EXECUTANDO) deve ser preemptado após o término do quantum.

- Gerenciamento de Estados:
    - Processos bloqueados (BLOQUEADO) não devem ser escalonados.
    - Processos terminados (TERMINADO) devem ser removidos da fila.

- Interação com o Usuário:
    - Exibir o estado da fila de prontos a cada ciclo.
    - Permitir a criação dinâmica de processos (ex: 30% de chance por ciclo).

5. Entregáveis

- Arquivos .c completos:
  - process.c
  - scheduler.c
  - rr_scheduler.c
  - main.c
- ***Não modificar os arquivos .h fornecidos.***
-

6. Critérios de Avaliação

1. **Corretude (60%):**
   - A fila circular funciona conforme especificado.
   - Preempção ocorre após o término do quantum.
2. **Organização do Código (25%):**
   - Ausência de vazamentos de memória.
   - Código modular e bem estruturado.
3. **Documentação (15%):**
   - Comentários explicando a lógica das funções críticas.

7. Dicas de Implementação

- **Fila Circular:** Use um ponteiro para o último processo da fila (*Scheduler->ready_queue*).
- **Exemplo de add_to_ready_queue:**

```
1. void add_to_ready_queue(Scheduler *sched, Process *p) {
2.     if (sched->ready_queue == NULL) {
3.         sched->ready_queue = p;
4.         p->next = p;
5.     } else {
6.         p->next = sched->ready_queue->next;
7.         sched->ready_queue->next = p;
8.         sched->ready_queue = p;
9.     }
10. }
11.
```

8. Exemplo de Saída:

```
=== Ciclo 1 ===

Processo 1 criado.

Executando: PID: 1 | Estado: EXECUTANDO | CPU: 1 | Restante: 4 | Quantum: 3

Fila de prontos:

PID: 1 | Estado: EXECUTANDO | CPU: 1 | Restante: 4 | Quantum: 3
```

9. Notas Finais

- Proibido usar bibliotecas externas além dos padrões do C (ex: stdio.h, stdlib.h).
- Dúvidas sobre as especificações devem ser esclarecidas com o professor.

Bom trabalho!