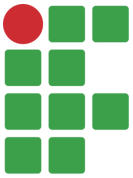

INSTITUTO FEDERAL
Triângulo Mineiro

Escalonamento: Simulação Round-Robin

Ranielly Ferreira dos Santos

Paulo Junior Rodrigues

Jullia Karolina De Paula

Patrocínio MG, Agosto 2024

Relatório Técnico

C++ Simulação de Escalonamento Round-Robin com Lista Encadeada

Introdução

No código fornecido, um simulador de escalonamento de processos Round-Robin foi escrito no C++ utilizando listas encadeadas para chegar a um. Exemplificado, o programa faz uso de processos que são ordenados pela sua chegada tempo e simulam a execução no round robin com um tempo de quantum definido pelo usuário. Esse relatório visa descrever a estrutura do código fonte, a lógica do jeito que ele procede, assim como considerações de como o programa agirá.

Estrutura do código

Declaração de bibliotecas

Para executar o código são utilizadas as seguintes bibliotecas :

<stdio.h> e <stdlib.h>: Bibliotecas de entrada e saída estilo C.
<string>: Utilizada para a manipulação de Strings.
<iostream>: Para entrada e saída estilo C++.

A junção das bibliotecas acima tornam a manipulação de dados, manutenção da memória, elevado nível de interação.

Estrutura No

A estrutura No, como dito, representa um processo, armazenando todos os dados necessários para a simulação. Podem-se citar os seguintes atributos da estrutura:

```
// Estrutura de n  para representar um processo
struct No {
    string nomeProcesso;
    int tempoChegada;
    int tempoUso;
    int tempoRestante;
    string estado;
    No* proximo;

    No(const string& _nome, int _chegada, int _uso) {
        nomeProcesso = _nome;
        tempoChegada = _chegada;
        tempoUso = _uso;
        tempoRestante = tempoUso;
        estado = "Pronto";
        proximo = NULL;
    }
};
```

nomeProcesso: nome do processo (string).
tempoChegada: tempo de chegada do processo (inteiro).
tempoUso: tempo total de uso do processo (inteiro).
tempoRestante: tempo disponível para que o processo seja concluído.
estado: Estado do processo definido inicialmente como “Pronto”.
próximo: ponteiro para o próximo nó em uma Lista encadeada.

O construtor permite já inicializar os atributos no momento da criação de novos processos passando o valor pelo parâmetro do construtor.

Estrutura Lista

A Lista serve para organizar os processos numa lista encadeada que representa uma fila de execução. As suas atribuições são:

primeiro: Ponteiro para o primeiro nó da lista.
ultimo: Ponteiro para o último nó da lista.

As suas funções principais é:

Construtor: princípio primário, que atribui os ponteiros primeiro e ultimo à posição null.

adicionar: adiciona novos processos no final da lista.

vazia: Checa se a lista estiver vazia.

ordenar Por Chegada : reordena os nós da lista respectivo tempoC chegada em ordem crescente através do algoritmo de inserção.

imprimirLista : Imprime a lista com os processos organizados.

Lógica do Programa

<u>Inserção dos Processos</u>

Os processos são inseridos na lista encadeada em função dos parâmetros do usuário fornecidos: nome, tempo de chegada e uso do tempo impressionam. Primeiro a adicionar uma lista cria um novo nó e insere ele ao final da lista:

Ordem por Tempo de Chegada

```
// Ordena os processos por tempo de chegada (Insertion Sort)
void ordenarPorChegada() {
    if (vazia() || !primeiro->proximo) return;

    No* ordenada = NULL; // Lista auxiliar para manter os n�s ordenados
    No* atual = primeiro;

    while (atual != NULL) {
        No* proximo = atual->proximo; // Armazena o pr�ximo n�
        if (!ordenada || atual->tempoChegada < ordenada->tempoChegada) {
            atual->proximo = ordenada;
            ordenada = atual;
        } else {
            No* aux = ordenada;
            while (aux->proximo && aux->proximo->tempoChegada <= atual->tempoChegada) {
                aux = aux->proximo;
            }
            atual->proximo = aux->proximo;
            aux->proximo = atual;
        }
        atual = proximo;
    }

    // Atualiza o ponteiro da lista para a nova lista ordenada
    primeiro = ordenada;
    // Atualiza o ponteiro "ultimo"
    ultimo = primeiro;
    while (ultimo && ultimo->proximo) {
        ultimo = ultimo->proximo;
    }
}
```

A função ordenarPorChegada faz com que os processos da lista encadeada sejam inseridos em ordem crescente de tempoChegada. Ela mostra o uso de :

Uma lista auxiliar (ordenada) que deve ser mantida para armazenar os nós necessariamente “por ordem”. Loop que passa pelos nós originais, inserindo-os em sua “posição certa” na lista auxiliar. Após a ordenação, lista e justaposição fica assim, os ponteiros primeiro e ultimo .

Simulação Round-Robin

roundRobin realiza a lógica que serve para executar cada nó por um intervalo de tempo(quantum) em determinado ou o tempo do processo for zero. A lógica também apresenta:

verificação do estado dos processos e ajuste do tempo atual; Atualiza-se o tempo de tempo restante aos processos; Registre a ordem de execução dos processos para posterior impressão.

```cpp
// Fun��o para simular o algoritmo Round-Robin
void roundRobin(Lista& lista, int quantum) {
    int tempoAtual = 0;
    int processosRestantes = 0;
    No* atual = lista.primeiro;

    // Array para registrar a ordem de execu��o
    string ordemExecucao[1000];
    int execIndex = 0;

    // Conta o n�mero de processos
    while (atual) {
        processosRestantes++;
        atual = atual->proximo;
    }
```

```cpp
 while (processosRestantes > 0) {
    atual = lista.primeiro;

    while (atual) {
        if (atual->tempoRestante > 0 && atual->tempoChegada <= tempoAtual) {
            int tempoInicio = tempoAtual;

            // Processar o quantum ou at� o tempo restante
            int tempoProcessado = (atual->tempoRestante < quantum) ? atual->tempoRestante : quantum;
            tempoAtual += tempoProcessado;
            atual->tempoRestante -= tempoProcessado;

            printf("Tempo %d-%d: %s executando\n", tempoInicio, tempoAtual, atual->nomeProcesso.c_str());

            // Registrar o processo na ordem de execu��o
            for (int i = 0; i < tempoProcessado; i++) {
                ordemExecucao[execIndex++] = atual->nomeProcesso;
            }

            // Verificar se o processo foi finalizado
            if (atual->tempoRestante == 0) {
                processosRestantes--;
                printf(" -> %s concluido no tempo %d\n", atual->nomeProcesso.c_str(), tempoAtual);
            }
        }
        atual = atual->proximo;
    }
```

```cpp
// Avan�ar tempo se nenhum processo estiver pronto
if (processosRestantes > 0) {
    int allDelayed = 1;
    atual = lista.primeiro;
    while (atual) {
        if (atual->tempoRestante > 0 && atual->tempoChegada <= tempoAtual) {
            allDelayed = 0;
            break;
        }
        atual = atual->proximo;
    }
    if (allDelayed) tempoAtual++;
}
```

Funcionamento do Programa

Entrada de Dados

esse programa solicita do usuário:

```
int main() {
    Lista lista;
    int quantProcessos, quantum;
    printf("=================================================\n");
    printf("    ALGORITMO DE ESCALONAMENTO ROUND-ROBIN (RR)   \n");
    printf("=================================================\n\n");

    printf("Qual a quantidade de processos (maximo 15): ");
    scanf("%d", &quantProcessos);
    if (quantProcessos > 15) {
        printf("Numero de processos deve ser no maximo 15.\n");
        return 1;
    }

    printf("Digite o tempo de Quantum: ");
    scanf("%d", &quantum);

    for (int i = 0; i < quantProcessos; i++) {
        char nome[100];
        int chegada, uso;

        printf("\n======= Cadastro do Processo %d ============\n", i + 1);
        printf("Digite o nome do processo: ");
        scanf("%s", nome);
        printf("Digite o tempo de chegada do processo: ");
        scanf("%d", &chegada);
        printf("Digite o tempo de uso do processo na CPU: ");
        scanf("%d", &uso);
        printf("=========================================\n");

        lista.adicionar(string(nome), chegada, uso);
    }
```

Número de processos máximo de 15. Tempo de quantum. O usuário fornece ao programa também:

Nome do processo. Tempo de controle.
Estrutura Principal (main)
O programa principal inicia e liga ao main numa seguinte instância de Lista:

Leitura de entradas de dados de parte do usuário.
Insere os processos “na” lista.
Ordena a lista em de tempo de chegada.
Exibe os processos ordenados.
Executa o algoritmo Round-Robin.

Verificação de resultados

Depois da simulação finalizada, esse programa exibe:

Retorna os detalhes sobre tempo de execução dos processos e a ordem dos processos.

Casos TESTE

teste 1

PROCESSO	TEMPO CHEGADA	TEMPO PROCESSAMENTO	
A	0	5	
B	0	8	
C	1	2	
D	2	5	**Quantum = 2**
E	2	4	
F	4	2	
G	5	6	

saída

```
A A B B C C D D E E F F G G A A B B D D E E G G A B B D G G B B
```

teste 2

PROCESSO	TEMPO CHEGADA	TEMPO PROCESSAMENTO	
A	0	5	
B	0	8	
C	1	2	
D	2	5	**Quantum = 3**
E	2	4	
F	4	2	
G	5	6	

Saída

```
A A A B B B C C D D D E E E F F G G G A A B B B D D E G G G B B
```

Conclusão

O programa apresenta uma simulação funcional do algoritmo Round-Robin, com o uso de listas encadeadas para armazenar os processos. De forma que o código seja modular, da mesma forma que faz a separação clara entre o código de No e Lista, aumentando o nível de legibilidade e manutenibilidade.