INSTITUTO FEDERAL
Amazonas | Campus Manaus Zona Leste

Albert França Josuá Costa
Curso:_____
Disciplina:_____
Nome:_____
Data:_____

LISTA 02

a) Seja pontual; b) Utilize caneta; c) Seja legível e d) Seja organizado.

1. Implemente na Linguagem C, utilizando ponteiros, os algoritmos de escalonamento de processos.

   O processo deve ter a seguinte estrutura (cuidado: Isso é um requisito funcional).

   - Identificador
   - Quantum
   - Tempo
   - Prioridade
   - Estado

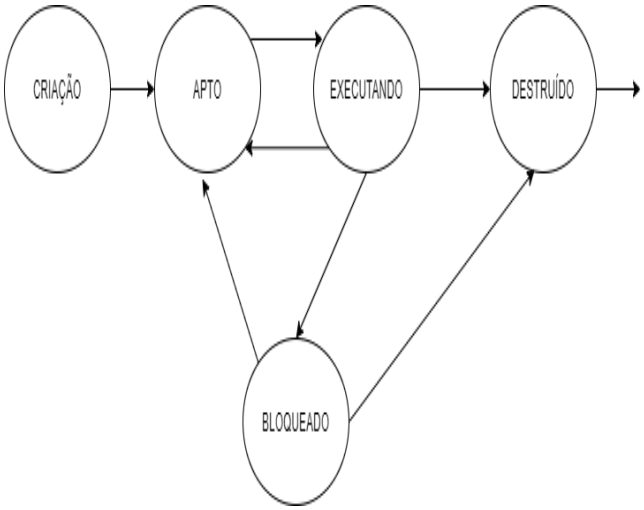
   (a) Algoritmo de Escalonamento *FIFO*.

   (b) Algoritmo de Escalonamento *Shortest-Job-First*.

   (c) Algoritmo de Escalonamento Circular *Round Robin*.

   (d) Algoritmo de Escalonamento por Prioridade.

   Para cada algoritmo de escalonamento, o seguinte cenário tem que ser implementado. Cada algoritmo de escalonamento será uma simulação diferente.



   O usuário (eu) irá criar n processos (pode e deve utilizar o código do trabalho anterior).

   No momento da criação dos processos, utilize as funções fornecidas, quando couber, para determinar o quantum, tempo e a prioridade.

   O quantum será definido em segundos.

O tempo será definido em segundos.

A prioridade será definida em números inteiros na ordem crescente (iniciando em 1).

SOMENTE após a criação dos n que a aplicação desenvolvida deverá entrar no modo de simulação.

Quando um processo estiver no estado executando ele deve simular isso escrevendo em um arquivo de texto a seguinte frase ("Processo" Identificador).

No arquivo de texto, as entradas devem ser inseridas sempre em uma nova linha.

Cuidado para não sobreescrever o arquivo.

Quando couber utilize a função fornecida (chamadaSistema) para verificar se o processo que está no ciclo de processador realizou um chamada de sistema.

Quando couber, utilize a função fornecida (desbloqueio) para verificar se o processo que solicitou uma operação de entrada e saída terminou a operação.

De acordo com o algoritmo de escalonamento definido na simulação, um processo que está no ciclo de processador e não realizou uma chamada de sistema deve retornar a fila de aptos.

Quando couber, utilize a função fornecida (terminar) para verificar se um processo que está no ciclo de processador terminou e vai ser destruído.

A cada iteração da simulação forneça ao usuário (eu) a opção de imprimir o diagrama de estados.

Uma operação de entrada e saída pode terminar em sucesso ou em falha.

A simulação deve encerrar quando todos os processos forem para o estado destruído.

É obrigatório manter a árvore de relacionamento de processos implementada no trabalho anterior. O código do trabalho anterior pode ser utilizado como ponto de partida.

FUNÇÕES FORNECIDAS

- chamadaSistema.
- desbloqueio.
- terminar.
- determinarQuantum.
- determinarTempo.
- determinarPrioridade.

ENTREGÁVEIS

- Código fonte.
- Executável.
- O arquivo .txt que simula o ciclo de processador.

- Relatório (4 páginas).

Relatório

- Introdução: Tem abordar os conceitos dos algoritmos de escolamento. (+- 1 página).
- Atividade Desenvolvida: Descrever o que foi feito.
- Resultados e Conclusão: Os resultados e as conclusões.
- Referência.

No máximo duplas

Data 26/11/2021 SIGAA

```c
/*
* Esses include's são necessários.
*/
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
/* Função responsável por definir se um processo que está no ciclo de processador
* realizou uma chamada de sistema.
* Retorno:
* 0 não realizou uma chamada de sistema.
* 1 realizou uma chamada de sistema.
*
*/
int chamadaSistema(){
    return rand()% 2;
}
/* Função responsável por definir se um processo que está no estado bloqueado
* terminou, ou não, a operação de entrada e saída com sucesso ou falha.
* Retorno:
* 0 Operação de entrada e saída finalizada com sucesso (O processo deve retornar a fila de aptos).
* 1 Operação de entrada e saída finalizada com falha (O processo deve ir para o estado Destruído).
* 2 Operação de entrada e saída não finalizada ainda (O processo deve continuar no estado bloqueado).
*/
int desbloqueio(){
    return rand()% 3;
}
/* Função responsável por determinar o quantum de um processo.
* Retorno:
* O inteiro retornado indica o tempo em segundos.
*/
int determinarQuantum(){
    return (rand()% 10)+15;
}
/* Função responsável por determinar o tempo de um processo.
* Retorno:
* O inteiro retornado indica o tempo em segundos.
*/
int determinarTempo(){
    return (rand()% 10)+15;
}
/* Função responsável por determinar a prioridade de um processo.
* O inteiro retornado indica a prioridade do processo em ordem crescente.
* 1<2<3<4...
*/
int determinarPrioridade(){
    return (rand()% 10)+1;
}
```