GERENTE DE PROCESSOS

Feito por: Zeus Moreira de Lima Pereira.

Data: 05/10/2020.

1. Resolução

A ideia de resolução para o programa "Gerente de processos" é baseada nos comandos necessários para o funcionamento do programa. Ao recebermos uma string (um comando) devemos avalia-la e desenvolver uma função de acordo com a necessidade do comando e a demanda que ele gera.

2. Funcionamento do algoritmo

Ao recebermos a estrutura de dados dada pela proposta do problema que nos fornece o horário de chegada de um processo e suas informações devemos tratar esses dados para solucionar o enunciado.

2.1) Adicionar o processo ao gerenciador: Devemos ler a instrução *"add"*, após isso devemos receber a prioridade, a hora de início e sua descrição, todos esses dados são adicionados na variável *"inicio"* do tipo *"celula"* que é definido como os dados do processo, logo depois esse processo é adicionado, pela função ***"add()"*** , ao vetor de registros dito como *"vet"* do tipo *"celula"*, e cada posição desse vetor carrega as informações sobre uma tarefa.

2.2) Executar o processo: Após o processo ser adicionado temos como opção executar o processo de maior prioridade ou de menor horário de chegada, para isso recebemos a instrução *"exec"* que após ser lida recebe o tipo de critério, prioridade ou tempo, dada pela variável *"exe"* que pode carregar apenas dois tipos de string, sendo elas *"-p"* e *"-t"*, representando a maior prioridade ou menor tempo respectivamente, que são definidas pela função ***"mergesort()"***. Depois disso, a tarefa selecionada será executada e excluída do gerenciador pela função ***"exec()".***

2.3) Próxima tarefa: Temos como uma opção saber qual será o próximo processo da lista de acordo com a prioridade ou tempo. Para isso recebemos a instrução *"next"*, a função ***"mergesort()"*** ordena os processos de acordo com a variável *"exe"* e a função ***"next()"*** mostra ao usuário a próxima tarefa desejada.

2.4) Mudar a prioridade de um processo: Outra alternativa é mudar a prioridade. Após recebermos a instrução *"change" recebemos* também a variável *"exe"* e outros dois parâmetros, dois inteiros se a variável *"exe"* conter *"-p"* e duas variáveis do tipo *"horario"* se *"exe"* conter *"-t"*. Com isso a função ***"change()"*** ou a função ***"change2()"*** atribuem ao processo o novo horário ou a nova prioridade procurando o processo que contém aquela determinada prioridade.

2.5) Mostrar a lista de processos de maneira ordenada: Essa parte se parece muito com a parte 2.3) acima, porém ao invés de mostrar o próximo processo mostramos a lista completa de prioridades de acordo com o tempo ou prioridade utilizando ***"mergesort()"*** para ordenar e ***"print()"*** para mostrar.

3. Funções primárias

3.1) Função "add()" :

- Entrada: *"celula inicio"* – Informações sobre o processo.
  "celula vet[]" – Manipula o processo.
  "int tamanho" – Espaço do vetor vet.

- Saida: Apesar de não retornar e não imprimir a função adiciona um dado a um vetor de registros para que futuramente o dado seja utilizado por outras funções.

3.2) Função "exec()" :

- Entrada: *"char exe[3]*" – Indicador de prioridade, onde carrega *"-p"* ou *"-t"*.
  "celula vet[]" – Manipula o processo.
  "int tamanho" – Espaço do vetor vet.
  "int p" – Primeira posição do vetor vet, útil para chamada do mergesort().

- Saida: Apesar de não retornar e não imprimir a função remove um elemento do vetor e o deixa ordenado.

3.3) Função "next()" :

- Entrada: *"char exe[3]*" – Indicador de prioridade, onde carrega *"-p"* ou *"-t"*.
  "celula vet[]" – Manipula o processo.
  "int tamanho" – Espaço do vetor vet.
  "int p" – Primeira posição do vetor vet, útil para chamada do mergesort().

- Saida: Mostra ao usuário qual a próxima tarefa a ser executada de acordo com indicador de prioridade.

3.4) Função "change()" :

- Entrada: *"celula vet[]"* – Manipula o processo.
  "int tamanho" – Espaço do vetor vet.
  "int novo" – Um inteiro que representa a nova prioridade do processo.
  "int anterior" – Posição que representa a prioridadr anterior do processo.

- Saida: Apesar de não retornar e não imprimir a função muda a prioridade de um elemento fazendo com que as outras funções utilizem dessa configuração.

3.5) Função "change2()" :

- Entrada: *"celula vet[]"* – Manipula o processo.
  "int tamanho" – Espaço do vetor vet.
  "horario nova" – Um inteiro que representa a nova prioridade do processo.
  "horario antes" – Posição que representa a prioridadr anterior do processo.

- Saida: Apesar de não retornar e não imprimir a função muda a prioridade de um elemento fazendo com que as outras funções utilizem dessa configuração.

3.6) Função "print()" :

- Entrada: *"celula vet[]"* – Manipula o processo.
  "int tamanho" – Espaço do vetor vet.
  "char exec[3]" – Indicador de prioridade, onde carrega *"-p"* ou *"-t"*.
  "int p" – Primeira posição do vetor vet, útil para chamada do mergesort().
- Saida: Mostra ao usuário a lista ordenada de acordo com o indicador de prioridade.

4. Funções elementares e a Main().

4.1) Função "converte_segundos()" :

- Entrada: *"celula vet[]"* – Manipula o processo.
  "int y" – O índice do elemento do vetor em forma de horario que necessitamos converter para comparação entre horas.

- Saida: Apesar de não retornar nem imprimir a função devolve o horario em segundos para facilitar a comparação para a função intercala.

4.2) Função "intercala()" :

- Entrada: *"celula vet[]"* – Manipula o processo.
  "int tamanho" – Espaço do vetor vet.
  "int q" – Indica a metade do comprimeto do vetor vet.
  "int p" – Primeira posição do vetor vet, útil para chamada do mergesort().
  "char exe[3]" - Indicador de prioridade, onde carrega *"-p"* ou *"-t"*.

- Saida: Retorna para a função *"**mergesort()"*** o vetor vet ordenado.

4.3) Função "mergesort()" :

- Entrada: *"int tamanho"* – Espaço do vetor vet.
  "celula vet[]" – Manipula o processo.
  "char exe[3]" - Indicador de prioridade, onde carrega *"-p"* ou *"-t"*.
  "int p" – Primeira posição do vetor vet, útil para a ordenação.

- Saida: Como ela é uma função utilitária, devolve o vetor ordenado para as funções que a chama dependendo do indicador de prioridade.

4.4) Função "main()" :

- Entrada: A entrada da função "***main()"*** se baseia em um primeiro momento de ler uma string definida por *"instrução"* que recebe um comando, dependendo desse comando pode-se haver uma outra leitura caracterizada, se a *"instrucao"* for *"add"* por exemplo a próxima leitura se baseia nos dados do processo, a leitura no geral é baseada no comando dito pelo usuário e com isso,também, as funções são invocadas para dar prosseguimento.

- Saida: As únicas saídas do programa são pertencentes ao comando “print” e “next”, logo quem executa as saídas são as funções ***“print()”*** e ***“next()”***