SCC0201 - Introdução à Ciência de Computação II

Trabalho 1

Professor: Diego Raphael Amancio

Estagiárias PAE:
Ana Caroline Medeiros Brito

DATA LIMITE: 28/10/2024 12hrs

Entrega no **run.codes**

Dúvidas:
carol_mb@usp.br

Gerente de processos

Um sistema operacional multitarefa deve manter o sistema em pleno funcionamento, atendendo demandas de todos os usuários a todo momento e, em particular, fornecendo a ilusão que o número de processos em execução simultânea é maior que o número de processadores do computador. Cada processo recebe uma fatia do tempo e a alternância entre vários processos se dá de forma rápida o suficiente para que o(a) usuário(a) acabe por acreditar que sua execução é simultânea. O sistema operacional usa alguns algoritmos para determinar qual processo será executado em determinado momento e por quanto tempo. Requisições de processos chegam a todo instante e devem ser atendidas de acordo com seu horário de chegada ou com sua prioridade.

Para execução de um processo, seu horário de chegada é uma importante característica levada em conta pelo sistema operacional. Além disso, como há tipos de usuários distintos no sistema, alguns processos associados a alguns usuários são mais prioritários que outros e esta é outra característica que o sistema operacional leva em conta na execução dos processos.

Tarefa

Sua tarefa é simular um gerenciador de processos. Seu programa deve ajudar o sistema operacional a gerenciar bem e de forma eficiente os processos, lidando com os seguintes comandos:

`add prior tempo descrição`: adiciona um processo à lista de processos a serem executados, onde prior é um inteiro positivo (prioridade) entre 1 e 99, tempo é um horário no formato **hh:mm:ss** e descrição é uma cadeira de caracteres com tamanho máximo 50

`exec opção`: executa um processo com a opção:

- `-p`: executa o processo com maior prior
- `-t`: executa o processo com menor tempo

`next opção`: mostra um processo de acordo com a opção:

`-p`: mostra em uma linha todas as informações do processo com maior prioridade, isto é, prior, tempo e descrição (cada campo separado por um espaço)

`-t`: mostra em uma linha todas as informações do processo com menor horário, ou seja, prior, tempo e descrição do processo

- `change opção anterior:novo`: modifica informações de um processo de acordo com a opção
  - `-p anterior:novo`: altera o campo prior com valor *anterior* para o valor *novo*
  - `-t anterior:novo`: altera o campo tempo com valor *anterior* para o valor *novo*, utilizando o formato **hh:mm:ss**
- `print opção`: imprime todos os processos a serem executados de acordo com a opção:
  - `-p`: imprime os processos em ordem decrescente de prioridade, onde as informações de cada processo são impressas em uma única linha (prior, tempo e descrição)
  - `-t`: imprime os processos em ordem crescente de horários, onde as informações de cada processo são impressas em uma única linha (prior, tempo e descrição)
- `quit`: termina a execução do gerente de processos

Considere que as prioridades e os horários são únicos.

Entrada

A entrada é composta por várias linhas, onde cada linha contém uma instrução para o gerente de processos.

Exemplo:

```
add 9 11:05:41 firefox
add 14 06:15:02 openoffice
add 5 05:26:18 xterm
add 7 10:44:34 emacs
add 16 05:43:21 gdb
add 11 22:47:56 garbage-collector
add 8 08:06:09 xfig
add 12 06:21:59 bash
add 13 04:11:20 nautilus
add 6 04:37:34 gnome
add 4 16:19:47 gcalc
add 17 22:40:32 x-session
add 1 07:33:25 printer-deamon
add 3 03:53:17 gimp
next -p
next -t
exec -p
exec -p
change -t 06:21:59|02:22:19
change -p 6|10
exec -t
print -p
print -t
quit
```

Saída

A saída depende das instruções de entrada. Observe que apenas as instruções `next` e `print` geram saída, sempre que uma delas for executada, imprima uma linha em branco no final. Em cada instrução, se for necessário imprimir um número inteiro (uma prioridade ou os componentes de um horário), imprima-o com duas casa decimais e com um 0 (zero) à esquerda, quando for o caso.

Exemplo:

```
17 22:40:32 x-session

03 03:53:17 gimp

14 06:15:02 openoffice
13 04:11:20 nautilus
11 22:47:56 garbage-collector
10 04:37:34 gnome
09 11:05:41 firefox
08 08:06:09 xfig
07 10:44:34 emacs
05 05:26:18 xterm
04 16:19:47 gcalc
03 03:53:17 gimp
01 07:33:25 printer-deamon

03 03:53:17 gimp
13 04:11:20 nautilus
10 04:37:34 gnome
05 05:26:18 xterm
14 06:15:02 openoffice
01 07:33:25 printer-deamon
08 08:06:09 xfig
07 10:44:34 emacs
09 11:05:41 firefox
04 16:19:47 gcalc
11 22:47:56 garbage-collector
```

Exigências

- As seguintes estruturas de dados deverão ser utilizadas em seu trabalho:

```c
/* Armazena um horário de chegada */

typedef struct {

    int hh;
    int mm;
    int ss;

} horario;

/* Armazena informações de um processo */

typedef struct {

    int prior;
    horario chegada;
    char descricao[MAX_DESCR];

} celula;
```

- Pelo menos um algoritmo mais eficiente como *Quick-sort*, *Merge-sort* ou busca binária deve ser implementado. Avalie a necessidade ou adequação com a estrutura de dados escolhida pelo grupo.

Entrega

- A solução deve ser desenvolvida em C. Verifique se o programa compila sem erros ou mensagens de alerta.
- A entrega será realizada pelo **run.codes**, será possível verificar se seu programa está se comportando de forma esperada através dos casos de testes disponíveis.
- Trata-se de um trabalho em grupo, basta que apenas um membro envie a solução no sistema.
- Pode assumir que não serão dadas entradas mal formatadas, por exemplo, prioridades negativas ou horários em formato diferente de **hh:mm:ss**.

Critérios de avaliação

O projeto será avaliado principalmente levando em consideração:

- Processamento correto de entradas e saídas do programa;
- Realização das tarefas descritas;
- Atender as exigências especificadas;
- Bom uso das técnicas de programação.