

Atividade Prática II

Gerente de processos

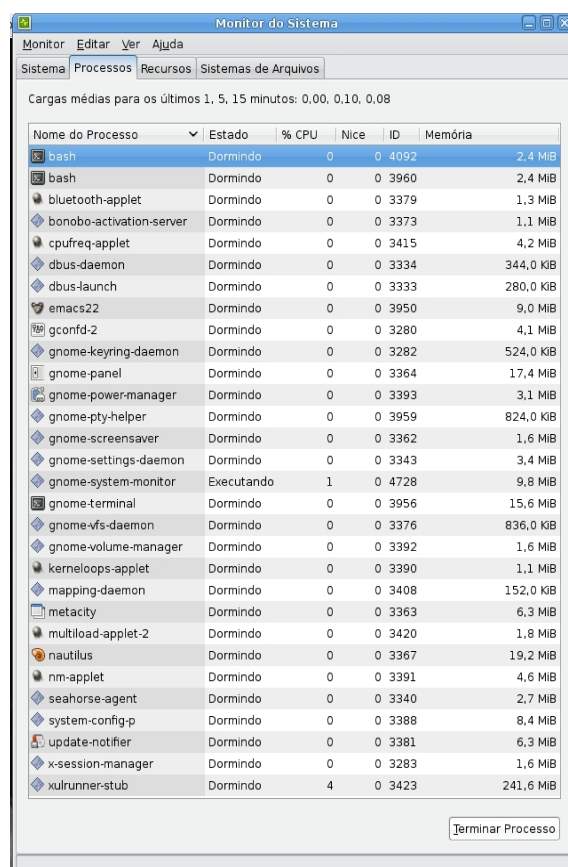
1 Descrição

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 a 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.

Para realizar todos esses objetivos, um gerenciador de processos ou de tarefas de um sistema operacional sempre dispõe de um conjunto de operações sobre processos. Dentre as mais importantes, podemos destacar as seguintes:

Para realizar todos esses objetivos, um gerenciador de processos ou de tarefas de um sistema operacional sempre dispõe de um conjunto de operações sobre processos. Dentre as mais importantes, podemos destacar as seguintes:



Nome do Processo	Estado	% CPU	Nice	ID	Memória
bash	Dormindo	0	0	4092	2,4 MIB
bash	Dormindo	0	0	3960	2,4 MIB
bluetooth-applet	Dormindo	0	0	3379	1,3 MIB
bonobo-activation-server	Dormindo	0	0	3373	1,1 MIB
cpufreq-applet	Dormindo	0	0	3415	4,2 MIB
dbus-daemon	Dormindo	0	0	3334	344,0 KB
dbus-launch	Dormindo	0	0	3333	280,0 KB
emacs22	Dormindo	0	0	3950	9,0 MIB
gconfd-2	Dormindo	0	0	3280	4,1 MIB
gnome-keyring-daemon	Dormindo	0	0	3282	524,0 KB
gnome-panel	Dormindo	0	0	3364	17,4 MIB
gnome-power-manager	Dormindo	0	0	3393	3,1 MIB
gnome-pty-helper	Dormindo	0	0	3959	824,0 KB
gnome-screensaver	Dormindo	0	0	3362	1,6 MIB
gnome-settings-daemon	Dormindo	0	0	3343	3,4 MIB
gnome-system-monitor	Executando	1	0	4728	9,8 MIB
gnome-terminal	Dormindo	0	0	3956	15,6 MIB
gnome-vfs-daemon	Dormindo	0	0	3376	836,0 KB
gnome-volume-manager	Dormindo	0	0	3392	1,6 MIB
kerneloops-applet	Dormindo	0	0	3390	1,1 MIB
mapping-daemon	Dormindo	0	0	3408	152,0 KB
metacity	Dormindo	0	0	3363	6,3 MIB
multiloop-applet-2	Dormindo	0	0	3420	1,8 MIB
nautilus	Dormindo	0	0	3367	19,2 MIB
nm-applet	Dormindo	0	0	3391	4,6 MIB
seahorse-agent	Dormindo	0	0	3340	2,7 MIB
system-config-p	Dormindo	0	0	3388	8,4 MIB
update-notifier	Dormindo	0	0	3381	6,3 MIB
x-session-manager	Dormindo	0	0	3283	1,6 MIB
xulrunner-stub	Dormindo	4	0	3423	241,6 MIB



add prioridade tempo descrição :

adiciona um processo à lista de processos a serem executados, onde **prioridade** é um número inteiro positivo, **tempo** é um horário de chegada no formato **hh:mm:ss** e **descrição** é uma cadeia de caracteres com no máximo 50 caracteres

exec opção :

executa um processo de acordo com a **opção**:

- p : executa o processo com maior prioridade
- t : executa o processo com menor horário de chegada

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, sua prioridade, seu horário de chegada e sua descrição
- t : mostra em uma linha todas as informações do processo com menor horário de chegada, ou seja, sua prioridade, seu horário de chegada e sua descrição

change opção anterior:novo(a) :

modifica a prioridade de um processo de acordo com **opção**:

- p **anterior|nova** : aumenta a prioridade **anterior** para prioridade **nova**, onde **anterior** e **nova** são números inteiros positivos
- t **anterior|novo** : diminui o horário de chegada **anterior** para o horário **novo**, onde os horários **anterior** e **novo** são fornecidos no 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 prioridades, onde todas as informações de cada processo (prioridade, horário de chegada e descrição) são impressas em uma única linha
- t : imprime os processos em ordem crescente de horários de chegada, onde todas as informações de cada processo (prioridade, horário de chegada e descrição) são impressas em uma única linha

quit

termina a execução do gerente de processos



Considere que as prioridades dos processos são distintas, assim como seus horários de chegada.

Sua tarefa nesta atividade é desenvolver um programa que ajude um sistema operacional a gerenciar bem as requisições de tarefas dos seus usuários, mantendo-o funcionando de maneira **eficiente**.

2 Entrada e saída

A entrada é composta por várias linhas, onde cada linha contém uma instrução para o gerente de processos, de acordo com a descrição das instruções dada na Seção 1. As prioridades são números inteiros entre 1 e 99. 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. Ainda, em cada uma dessas instruções, sempre que seja necessário imprimir um número inteiro (uma prioridade ou os componentes de um horário), imprima-o com duas casas decimais e com um 0 (zero) à esquerda, quando for o caso.



3 Exemplo de entrada

```
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-daemon
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
```



4 Exemplo de saída

```
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
```

5 Exigências

Nesta Seção destacamos alguns itens que são obrigatórios na execução da atividade. Caso um desses itens não sejam seguidos, a atividade não será considerada e receberá nota **ZERO**.

5.1 Estruturas de dados

Você **DEVE** usar a seguinte estrutura de dados para representar um processo na implementação de seu programa:



```
/* 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+1];  
} celula;
```

A estrutura de dados utilizada para organizar os processos fica a seu critério. Porém, pela descrição da atividade, a estrutura mais indicada para a solução do problema fica óbvia.

Lembre-se que estamos esperando um programa **eficiente**.

5.2 Organização do código-fonte

Você **DEVE** organizar seu código-fonte em 3 (três) arquivos:

1. Um arquivo com extensão **.c** contendo todas as funções de apoio à solução da atividade;
2. Um arquivo-cabeçalho com extensão **.h** referente ao arquivo anterior;
3. Um arquivo com extensão **.c** contendo somente a função principal;

Seus arquivos-fonte na linguagem C devem estar bem organizados. Um programa na linguagem C tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. Não esqueça da documentação de seu programa.

5.3 Compilação

Os professores usam o compilador da linguagem C da coleção de compiladores GNU **gcc**, com as opções de compilação **-Wall -ansi -pedantic** para corrigir os programas. Se você usar algum outro compilador para desenvolver seu programa, antes de entregá-lo



verifique se o seu programa tem extensão `.c`, compila sem mensagens de alerta e executa corretamente.

6 Entrega

Instruções para entrega da sua atividade:

1. Grupos e forma de entrega

A atividade pode ser realizada individualmente ou em grupos com no máximo dois integrantes (duplas). O trabalho deverá ser entregue no Ambiente Virtual de Aprendizagem da UFMS (AVA) **até as 23:59 do dia 29/11/2020**. No caso da atividade ser realizada em duplas, **apenas um** dos integrantes deve realizar a entrega. Lembre-se de colocar o nome dos integrantes nos arquivos-fonte.

A entrevista será feita em grupo e também individualmente. Ou seja, tanto o grupo quanto cada aluno será avaliado nas entrevistas. Desse modo, recomendamos que todos os alunos do grupo participem de todas as etapas do desenvolvimento. A ordem das entrevistas será sorteada e divulgada para todos por meio do AVA. As entrevistas ocorrerão no dia **dia 04/12/2020 a partir das 19:00**.

A nota final do trabalho é composta pela nota da implementação e nota da entrevista como na equação a seguir:

$$NotaTrabalho2 = \frac{Nota_{entrevista}}{10,0} * Nota_{implementacao}$$

2. Atrasos

Atividades atrasadas não serão aceitas. Não deixe para entregar sua atividade na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, e/ou falha de conexão com a internet, sugerimos que a entrega da atividade seja feita pelo menos um dia antes do prazo determinado.

3. Erros

Programas com erros de compilação receberão nota **ZERO**. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.

4. Verificação dos dados de entrada

Não se preocupe com a verificação dos dados de entrada do seu programa. Seu programa não precisa fazer consistência dos dados de entrada. Isto significa que se,



por exemplo, o seu programa pede um número entre 1 e 10 e o usuário digita um número negativo, uma letra, um cifrão, etc, o seu programa pode fazer qualquer coisa, como travar o computador ou encerrar a sua execução abruptamente com respostas erradas.

5. Conduta Ética

A atividade deve ser feita de acordo com o descrito no item 1 da Seção 6. Cada estudante tem responsabilidade sobre cópias de sua atividade, mesmo que parciais. Não faça a atividade em grupos maiores que o permitido e não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir ideias sobre a atividade, mas **NÃO** copie o programa! Trabalhos considerados plagiados terão nota **ZERO**.