Laboratório I – Sistemas Operacionais

Introdução: o objetivo deste projeto é introduzir o aluno nas facilidades de manipulação de processos no sistema operacional UNIX/LINUX. A implementação deverá ser feita em uma distribuição Linux e usando linguagem C.

Descrição:

O aluno deverá escrever um programa ***execute*** que receba outro comando como argumento e execute o comando. Por exemplo, na seguinte chamada do programa:

```
% execute cat /etc/motd
```

o programa vai chamar o comando cat no arquivo /etc/motd, que vai imprimir na tela a "*mensagem do dia*". Após ter completado a execução do comando especificado, o execute deverá mostrar algumas estatísticas dos recursos do sistema usados pelo comando. Em particular, o execute deverá mostrar:

Estatísticas do processo
1. O total de tempo de CPU utilizado (ambos: usuário e sistema) em milisegundos
2. O tempo de relógio ("wall-clock") gasto para executar o comando (em milisegundos)
3. O número de vezes que o processo foi interrompido involuntariamente (isto é, expirou a fatia de tempo disponível para ele e outro processo assumiu a CPU)
4. O número de vezes que o processo cedeu a CPU voluntariamente (por exemplo, esperando por um recurso)

Shell de comandos básicos

A conclusão satisfatória da parte descrita acima pelo programa corresponde a 70% da tarefa. Uma parcela de 20% do valor total da tarefa compreende estender o programa de forma a este se comportar como um interpretador de comandos (Shell) se não forem passados argumentos no momento de sua chamada. Ele deverá funcionar como antes, aceitando comandos digitados posteriormente. O programa deverá aguardar continuamente um comando (que pode ter vários argumentos separados por espaço em branco) e então executar o comando e mostrar o resultado na tela e imprimir as estatísticas. O trabalho envolve dividir a linha de texto que está sendo lida em uma lista de argumentos. O programa também deverá tratar também dois comandos embutidos (internos):

exit: encerra o interpretador de comandos

cddir: resulta na mudança do diretório para dir

O programa também deverá terminar se um fim-de-arquivo é detectado na entrada de dados. O aluno poderá assumir que uma linha não vai conter mais que 128 caracteres ou mais que 32 argumentos distintos. Um exemplo de uma sessão usando o programa é dado abaixo onde os comentários são dados entre "<>". O prompt do execute é indicado por "→" e o interpretador de comandos normal por "%"

Exemplo 1

```
% execute
  • cat /etc/motd
          < mostra na mensagem do dia>
          <mostra as estatísticas do comando cat>
  • cddir
          <o diretório corrente é alterado para dir>
          < não há estatísticas neste caso por ser um comando
          interno>
  • exit
% <de volta ao Shell do sistema>
```

Tarefas em background

Para obter os 10% finais da tarefa, o programa deve ser estendido para manusear tarefas em background. Uma tarefa em background é indicada pela colocação de um caracter "e comercial" ('&') ao final de uma linha de entrada. Quando uma tarefa executa em background, o seu interpretador de comandos não deverá esperar até que a tarefa se complete, mas imediatamente ficar disponível para que o usuário entre com outro comando. Note que qualquer saída resultante do comando em background vai direto para o terminal e vai intercalar com as saídas de seu Shell e de outros comandos. É preciso também acrescentar outro comando interno para o execute:

-**jobs** – lista todas as tarefas em background

Um exemplo de uma sessão é mostrada abaixo:

%execute

```
➔ numbercrunch&
   [1] 12345      <indica a tarefa em background e seu process id>
➔ jobs
   [1] 12345 numbercrunch<mostra o process id e o nome do comando para
a tarefa>
➔ ls             < lista os arquivos no diretório corrente>
                 < mostra as estatísticas deste comando>
➔ cat /etc/motd
   [1] 12345 Completed <indica  que  a  tarefa  em  background  está
concluída>
            < mostra estatísticas do comando numbercrunch>
            <mostra a mensagem do dia>
            <estatísticas do comando cat>
➔ exit
     %      < retorna para o Shell do sistema>
```

Algumas chamadas do sistema úteis para a implementação da atividade:

- fork() → cria um novo processo
- getusage() → obtém informação sobre utilização de recursos. Obs.: nem todas as versões do Unix/Linux preenchem todos os campos de retorno deste comando. Seu programa deverá mostrar simplesmente o que recebe como retorno.
- clock_gettime() → obtém informações específicas de tempo.
- gettimeofday() → obtém a hora atual para cálculo do relógio (wall-clock)
- execxx() → executa um arquivo (pesquisar as variacoes e usar o mais apropriado)
- wait() → espera pelo término de um processo.
- chdir() → muda o diretório corrente de um processo.

Para obter mais informações sobre estas rotinas, use o comando “man” do Unix/Linux. As páginas do manual são organizadas em sessões. Sessão 1 é para comandos do Unix, sessão 2 é para chamadas ao sistema e a sessão 3 é para rotinas de bibliotecas. Por exemplo, para obter informações sobre a chamada ao sistema wait(), use “man 2 wait” para exibir explicitamente uma determinada sessão.