UFC – Quixadá - QXD0013 – Sistemas Operacionais - Prof. Cristiano

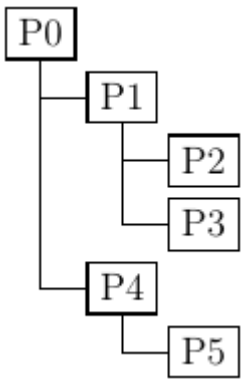
Instruções:
– Utilize o Linux para realizar a atividade. Sempre que estiver em dúvida sobre o funcionamento de uma função ou chamada, utilize o manual do Linux, diponível através do comando *man*.
– Submeta as respostas preenchendo o gabarito disponível no Moodle.

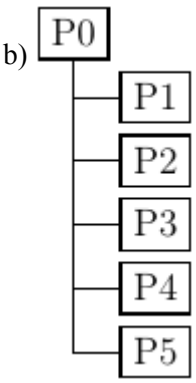
Exercício de laboratório 01 – Processos e Threads

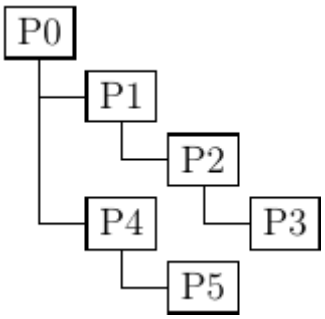
Questões

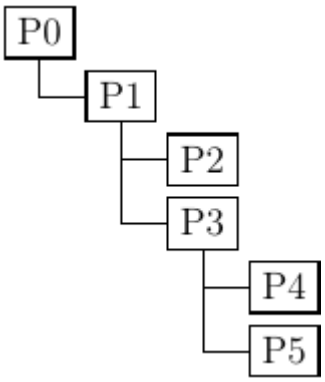
→ Compile e execute o código do aquivo *lab01a.c* algumas vezes e em seguida responda aos itens abaixo:

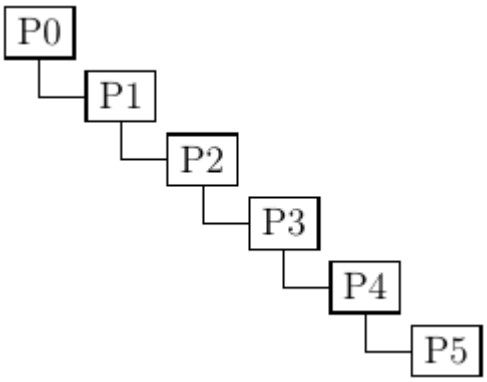
1) Qual das árvores de processos abaixo é similar à árvore gerada com a execução do código?

a)


b)


c)


d)


e)


2) Dadas as afirmações abaixo, temos que:

I. Para o S.O. a execução do código resulta na criação de 1 processo pai, 6 filhos, totalizando 6 threads.
II. Para o S.O. a execução do código resulta na criação de 1 processo pai, 5 filhos, totalizando 6 threads.
III. O processo pai criado possui 5 filhos, e, portando, tem 5 threads além da sua thread principal.
IV. O PID de um processo sempre será menor que o PID dos seus filhos

a) Somente I está correta
b) Somente II está correta
c) I e III estão corretas
d) II e III estão corretas
e) II, III e IV estão corretas

3) Considerando a execução do código, podemos afirmar que:

I. Todos os processos gerados são suspensos (bloqueados) pelos menos duas vezes.
II. No código, a presença da chamada *sleep(5)* na função *my_routine* garante que nenhum processo é escalonado duas vezes consecutivas
III. Todos os processos realizam pelo menos uma chamada *write*.
IV. Uma *thread* suspensa pela função *sleep(N)* pode voltar a executar antes de *N* segundos, caso algum sinal seja recebido pela *thread*.

a) Todas as afirmativas acima estão corretas.
b) Somente a afirmativa I está incorreta.
c) Somente a afirmativa II está incorreta.
d) Somente a afirmativa III está incorreta.
e) Somente a afirmativa IV está incorreta.

→ Compile e execute o código do aquivo *lab01b.c* , e visualize as informações das threads criadas.

Utilize o comando abaixo para compilar:

gcc -Wall lab01b.c -lpthread

Para visualizar as informações das threads criadas, abra uma nova aba ou janela do terminal durante a execução do código e digite o comando:

ps -eLf |grep -e PID -e <nome_do_executavel>

Exemplo:

ps -eLf |grep -e PID -e a.out

→ As questões a seguir referem-se ao código do aquivo *lab01b.c* . Antes de respondê-las , siga os passos a seguir (se preferir, faça um backup do arquivo e repita essas instruções algumas vezes):

- Na função ***thread_routine*** comente ou exclua a linha que contém a chamada ***sleep(5)***;
- Compile e execute o código **algumas vezes** e observe seu comportamento
- Na função ***thread_routine*** descomente a linha que contém a chamada ***sched_yield*()**;
- Compile e execute o código algumas vezes novamente e observe seu comportamento
- Descomente as três primeiras chamadas na função ***main***
- Compile e execute o código **algumas vezes** novamente e observe seu comportamento

Responda:

4) Em relação ao compartamento do processo, marque a alternativa CORRETA.

a) A ordem com que as mensagens são impressas pode ser garantida apenas pela função *sched_yield*.
b) A ordem com que as mensagens são impressas pode ser garantida apenas pela função *sleep*.
c) A ordem com que as mensagens são impressas pode ser garantida pela função *sched_yield* em conjunto com *sleep*.
d) A ordem com que as mensagens são impressas pode ser garantida apenas pela função *sched_setaffinity*.
e) A ordem com que as mensagens são impressas não pode ser pode ser garantida.

5) Sobre a função função *sched_yield*, é INCORRETO afirmar que:

a) É uma função que desabilita o núcleo do processador que está executando a *thread*
b) Seu objetivo é liberar o processador para ser usado por outra *thread*.
c) Por definição, retorna -1 em casdo de erro, mas no Linux sempre retorna 0.
d) Se usada indiscriminadamente pode diminuir o desempenho geral do sistema por provocar mudanças de contexto desnecessárias.
e) Se usada corretamente pode melhorar o desempenho do sistema.

6) Marque a opção CORRETA considerando a seguinte linha de código da função *main*:

status = pthread_create(&threads[i], NULL, thread_routine, &thread_data[i]);

a) Somente a i-ésima thread criada tem acesso a *threads[i]*
b) A função pthread_create não pode ser usada caso o código contenha uma chamada *fork.*
c) Tanto *thread_data*, quanto *threads* são arrays que poderiam ser acessados por todas as threads.
d) Substituir *thread_data[i]* por *i* não altera a saída do programa.
e) *pthread_create* é uma função exclusiva do Linux, sem equivalente em outros sistemas operacionais.

<u>Sugestão de problema (Não vale nota):</u>

– Crie um código com pelo menos dois processos, um pai e um filho, de forma que os processos se comportem da seguinte maneira:

- O processo pai deve configurar a prioridade do processo filho para ***p* + *1***, onde ***p*** é a prioridade do processo pai, sempre que o valor de ***p*** for alterado..
- Ambos os processos pai e filho devem escrever uma mensagem na tela sempre que tiverem suas prioridades alteradas.
- Ambos os processos devem encerrar quando sua prioridade for igual a -15 (após exibirem as mensagens).

Utilize o comando ***renice*** (ver manual com o comando ***man renice***) para alterar a prioridade do processo pai. Utilize as seguinte funções para obter ou modificar as prioridades:

```
Obter:     int getpriority(int which, id_t who);
Modificar: int setpriority(int which, id_t who, int prio);
```

- O parâmetro *which* deve receber o valor *PRIO_PROCESS*
- O parâmetro *who* deve receber o *pid* do processo
- O parâmetro *prio* deve receber algum valor modificador de prioridade no intervalo -20 a 19, sendo que, quanto mais baixo o valor, maior a prioridade.

➢ Para mais informações acesse o manual das funções digitando ***man setpriority*** ou ***man getpriority*** em um terminal Linux.
➢ Tente a execução dos comandos e/ou processos com privilégio de super usuário (como *root* ou usando *sudo*) caso das alterações nas prioridades não sejam efetivadas pelo S.O..