Oficina Linux --/--/----

Aula __: Gerenciamento de Processos

Professor: Eraldo Silveira e Silva eraldo@ifsc.edu.br

1 Objetivos

- Consolidar o conceito de processo;
- Usar o comando *ps* para verificar os processos do sistema;
- Encerrar, suspender e reativar processos suspensos;

2 Estados de Processos no Linux

Os estados possíveis de um processo no linux são:

- R - se executando (*running*) ou na fila de execução;
- D - dormindo na espera por algum evento;
- S - dormindo na espera por algum evento ou sinal (*interruptible sleep*);
- T - Parado(*stopped*) (por um sinal ou sendo depurado);
- A - processo Zombie (foi terminado mas ainda presente no sistema);

Uma forma de processos (ou do próprio sistema operacional) se comunicarem com outros processos é transmitir sinais para o mesmo. Existem vários tipos de sinais. Um dos mais famosos é o SIGKILL que "mata"(mas nem sempre) um processo. O sinal SIGSTOP para um processo e o sinal SIGCONT faz com que um processo parado continue a sua execução.

Para ver todos os processos do sistema pode-se fazer:

ps aux | more

Note a coluna PID. Ela indica o número identificador do processo. Em STAT pode-se observar o estado do processo. Existe uma relação de paternidade entre processos e é possível verificar uma árvore de processos com *pstree*. O comando *top* é uma espécie *ps* em tempo real.

3 Roteiro

1. Logar no sistema, na interface gráfica, como *aluno*;
2. Abrir um console de trabalho: Aplicações->Acessórios->Console
3. Verificar o terminal que você está:

   tty

> **Nota:** O comando tty mostra o nome do terminal que será associado a entrada padrão dos processos que sobre ele se executarem.

4. Executar o comando yes:

   yes Testando

   Nota: este comando, quando executado, se torna um processo cuja função é imprimir constantemente na saída padrão a cadeia de caracteres passada como parâmetro

5. Para encerrar o processo que está ocupando o terminal faça: CTRL-c ou (CTRL-c)

   Nota: Observe que o processo foi "morto". O CTRL-c envia um sinal SIGKILL para o processo.

6. Coloque novamente o processo em execução, mas agora em *background*:

   yes Testando > /dev/null &

7. Trazer o processo novamente para o terminal (colocar em *foreground*):

   fg

8. Suspenda o processo com: CTRL-Z ou (CTRL-SHIFT-z)

   Nota: Suspender o processo é parar o processo mas mantendo-o no sistema com todos os recursos menos a cpu.

9. Verifique o estado do processo supenso com:

   ps -l

10. Coloque em execução mais algumas instâncias do processo *yes*:

    yes Testando > /dev/null &

    yes Testando > /dev/null &

11. Neste momento existem 3 processos *yes*, sendo que dois em execução e um suspenso. Conferir os processos que você lançou com:

    ps u

    Nota: este comando lista todos os processos atrelados aos diversos terminais e que pertencem ao usuário. Note que o próprio *bash* aparece. Um para cada terminal

12. Abra duas abas de console e para cada uma delas verifique o terminal que se encontra:

    tty

13. Execute em cada uma delas (menos na primeira)um processo yes:

    yes Testando

14. Volte para o primeiro terminal e liste todos os processos que estão atrelados aos terminais, usando a forma longa:

    ps l

15. Faça também com comando *top* para verificar os processos em execução:

    top

16. Vamos "matar" um processo yes do terminal 2:

    kill -KILL <numero_pid>

    Nota: Cada processo tem um identificador: PID. O PID deve ser fornecido no comando *kill*. Note que o comando kill envia sinais para um processo.

17. Vamos usar o comando kill para suspender e continuar a execução do processo *yes* no terminal 3.

    kill -STOP <numero_pid>

18. Ver os processos:

    ps -l

    Nota: Observe o código STAT. Os processos em execução estão com R (running), os parados com T (stopped). Os processos dormindo com S (sleeping)

19. Coloque o processo em execução novamente:

    kill -CONT <numero_pid>

4 Alguns exercícios adicionais

1. Execute o firefox a partir da interface gráfica. Descubra o PID do processo firefox e depois "mate-o" a partir de um terminal;
2. O processo *init* é o pai de todos os processos. Qual é o PID deste processo? Tente matar o processo *init*.
3. Faça um comando usando *pipes* e *wc* para imprimir o número de processos pertencentes ao usuário *aluno*.
4. Execute o comando *sleep 10*. O que ele faz?
5. Execute um comando *sleep 3000* . Verifique e anote o PID deste processo. Faça um logout do sistema. Logue novamente e verifique se o o processo ainda está lá.
6. Execute o comando *sleep* com auxílio do comando *nohup*, da forma: *nohup sleep 3000* Anote o PID do preocesso, saia novamente da sessão e entre novamente no sistema. Verifique se o processo *sleep* está presente.
7. Coloque dois processos em background (por exemplo, *sleep 10 &; sleep 20 &*) e verifique os processos que estão em background usando o comando *jobs*.
8. Para executar dois ou mais comandos em uma mesma linha, separe-os com ";". Faça o seguinte comando e analise a saída:

   echo Teste 1; echo Teste2
9. Para executar em paralelo use a execução em background:

   yes &; yes&
10. Faça um *ps aux* e descubra quais são os três processos que mais consomem CPU. Faça uma pesquisa para descobrir quem são estes processos.
11. Ler os capítulos 6 e 7 da apostila. Fazer o resumo e enviar para o professor.