1º Laboratório SOA – Gerencia de memória/processos – 19/08/2010

As atividades de laboratório consistem em conhecer na prática os comandos do Sistema Operacional Linux relacionado ao gerenciamento de memória/processos.

Os registros das atividades deverão ser registrados em relatório individual ao final de cada aula.

Controlando processos

Quando o usuário solicita um aplicativo, o *shell* informa para o sistema operacional das intenções do usuário em utilizar determinado aplicativo. Então, é solicitado ao kernel que o mesmo seja localizado no computador. O principal meio de armazenamento para os computadores são os HD's (*Hard Disk*). Tão logo as informações solicitadas são encontradas, o HD as repassa para a memória de acesso aleatório (RAM). De posse das informações necessárias para executar o processo, a memória RAM repassa as informações ao processador, onde o mesmo irá adicionar dados necessários para a sua execução, tais como número referente à sua solicitação - que é diferente para cada pedido -, hora do pedido, nome do solicitante, entre outras informações.

As informações entre a RAM e o processador podem ser encontradas em vários estados, podendo o processo estar:

- Rodando no processador;
- Esperando por algum dado necessário para continuar sua execução
- Residente na memória
- Parado ou;
- Em depuração (por depuração entende-se o tratamento que é dado para as informações na tentativa de recuperá-las).

O conjunto de todas essas informações é chamado de *tabela de processos*.

Nessa tabela encontram-se todas as informações apresentadas acima, além de informar o estado do processo no computador.

Utilitários que manipulam essa tabela estão disponíveis no GNU/Linux. Dentre as possibilidades estão a suspensão dos processos em execução, a continuação de algum processo que estava suspenso ou mesmo a finalização precoce de algum processo mal comportado.

free

O comando exibe a quantidade de memória livre e utilizada pelo sistema.

Sintaxe:

```
$ free
```

Opções:

–t	Exibe os totais da memória RAM e SWAP
–m	Exibe os valores de saída em MB
–k	Exibe os valores de saída em KB
–mt	Exibe os valores de saída mais os totais em MB

ps

O comando ps exibe os processos se utilizando das informações existentes na tabela de processos. Dentre as opções disponíveis, estão aquelas que controlam os tipos de processos a serem exibidos, bem como os diferentes tipos de informações presentes.

Sintaxe:

```
ps [opções] [txx]
```

Opções:

a - apresenta os processos do console atual, independentemente do usuário que os executou;

c - apresenta somente o principal nome do processo;

e - apresenta as variáveis de ambiente a qual o processo foi solicitado;

f - apresenta o nome em estrutura de árvore;

l - formato longo - apresenta todas as informações da tabela de processos;

u - apresenta o resultado em formato de usuário, ou seja, mais amigável;

x - apresenta os processos independe de seu terminal.

Exemplo:

```
$ps a

PID TTY          STAT     TIME COMMAND

888 tty2         S        0:00 login -- root

1145 tty2         S        0:00 - bash
```

```
2480 tty2        R       0:00 ps a
```

No exemplo acima foi utilizada a opção ***a***, que listou as informações existentes na tabela de processos do console atual. Nesse caso o console foi inicializado pelo super-usuário, logo após, a inicialização do shell de comando utilizado (no exemplo é o bash) e o comando ps, que sempre gera um valor de processo na sua chamada.

Também é observável a apresentação de novas palavras, tais como PID STAT TIME e COMMAND, que são campos pertinentes à tabela de processos do usuário. Existem mais colunas, conforme apresentado abaixo:

PID - é um número único que o processo recebe ao solicitar o processador;

TTY - console que chamou o processo;

STAT - Informa a situação em que se encontra o processo, podendo ele estar em execução (R), aguardando (S), parado ou em depuração (T), em estado zumbi (Z), aguardando sem interrupção (D). Estas são as informações que pertencem ao primeiro campo, já no segundo podemos encontrar detalhes sobre páginas residentes na memória (W), e para o terceiro campo podemos ter prioridade positiva (N) em seu valor;

TIME - tempo atual de utilização do processador;

COMMAND - apresenta o nome do comando que foi solicitado;

USER - nome do usuário solicitante;

%CPU - porcentagem de uso da cpu;

%MEM - porcentagem de uso da memória;

RSS - quantidade de memória física utilizada durante o seu processo, o valor é apresentado em Kbytes;

START - hora do início do processo.

A visualização de todas essas colunas pode ser feita utilizando o comando ps aux.

Para maiores detalhes: man ps.

10.3 top

Muitas vezes somente o comando "ps" não é conveniente o bastante para se monitorar o estado dos processos de um sistema. Isto porque o "ps" lista as informações relativas ao momento de sua execução.

O comando "top" funciona de modo interativo, atualizando as informações em tempo real. O intervalo padrão de atualização das informações é de 5 segundos. As atualizações incluem novos processos que são finalizados, ou seja, que deixam de ser listados, e também as mudanças na quantidade utilizada de processador e memória.

Sintaxe:

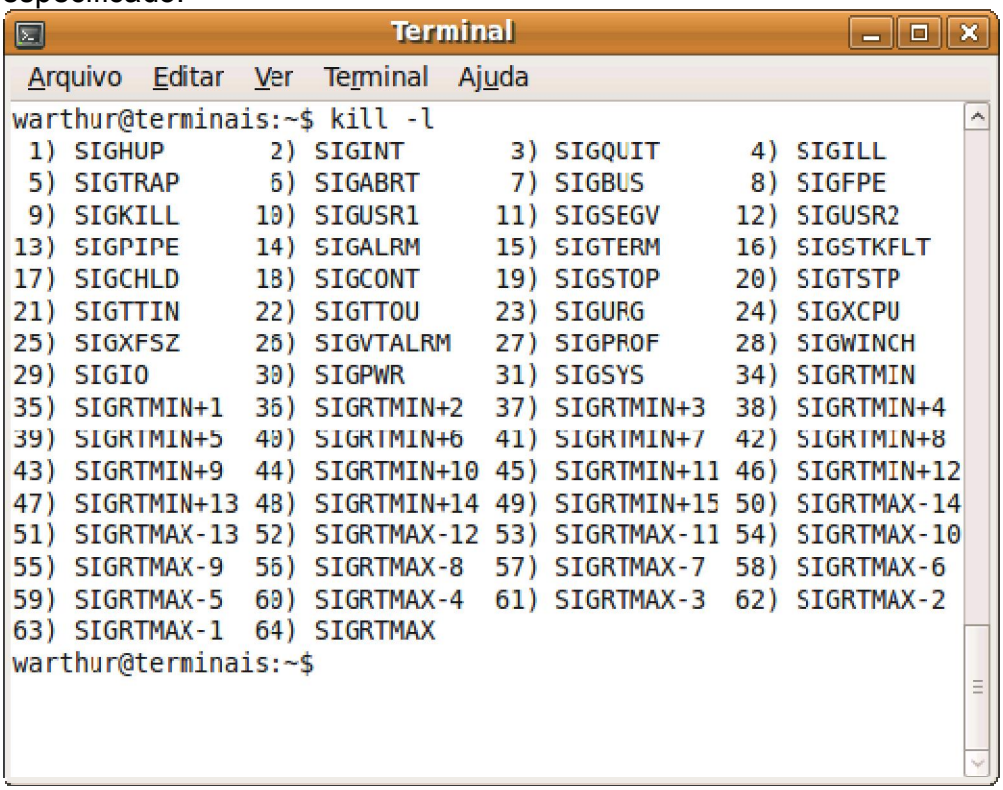
```
top
```

kill

Um programa pode sofrer vários tipos de intervenções, quando solicitado pelo usuário. Entre as principais destacam-se a atualização e retirada do processo.

Quando o usuário decide intervir na tabela de processos é o comando "kill" que deverá ser utilizado, ou seja, se um determinado programa demorar dar uma resposta o próprio usuário poderá solicitar a sua parada e retirada do processador.

À essa solicitação é dada o nome de sinal, que varia entre 64 opções para o usuário. Para listá-los basta utilizar o comando 'kill -l'. Por padrão o comando kill utiliza o sinal TERM (15) quando não especificado.



Sintaxe:

```
kill [opção] [pid]
```

Opção:

-l - lista todas as opções de sinais - tanto o nome do sinal quanto o seu valor numérico

-s - especifica o sinal a ser enviado de acordo com as opções listadas pelo comando -l

pid - número de identificação do processo, a qual é desejado enviar o sinal.

Comandos mais comuns listados com a opção -l:

matar o processo: -9

atualizar o processo: -1

Exemplo:

- antes de se utilizar o "kill" é necessário saber qual a identificação do processo em questão:

```
$ps auxc

USER    PID  %CPU %MEM VSZ   RSS  TTY STAT START  TIME COMMAND

...

aluno   1420 14.2 10.7 23088 14032 ?    S  0:00   0:02 netscape

...
```

Para este exemplo será encerrado o browser netscape a partir da sua identificação de processo.

Observação: veja detalhes das opções do comando "ps" no capítulo referente.

```
$kill -9 1420
```

logo após esse comando o netscape será fechado.

jobs

Jobs, ou tarefas, são processos que estão rodando no console atual, disparados pelo usuário atual. As tarefas, além de seus números de processo, também possuem um número próprio, utilizado como índice nos comandos de manipulação de tarefas, mencionados a seguir. O comando "job" limita-se a listar as tarefas existentes.

Sintaxe:

```
jobs [opção]
```

Opção:

-l - apresenta junto com o valor do "job" o valor de identificação do processo, que foi dado na chamada do programa para ser rodado em segundo plano.

Exemplo:

```
$jobs -l
```

```
[1]     1765 Running            netscape &
```

No exemplo dado, o valor do job foi apresentado entre colchetes. A coluna a seguir representa o valor do PID, A terceira coluna informa o estado do programa dentro do processador e a última, o nome do programa.

Controle de tarefas

Suponha que um usuário, logado em um console serial, esteja depurando uma determinada aplicação. Para isso, ele precisa fazer uso de um editor de texto e de um interpretador de comandos, a partir de onde a aplicação será executada e re-compilada.

Utilizando os recursos e comandos de tarefas, o usuário consegue colocar o editor de texto em segundo plano (geralmente através da combinação de teclas Ctrl+Z), para então recompilar a aplicação, e em seguida executá-la para testar as alterações feitas. Depois dessa primeira fase de testes, o usuário retorna ao editor de texto, faz novas alterações na aplicação e retorna ao interpretador de comandos para novos testes.

Sem os recursos de controle de tarefas, o usuário teria que fechar o editor de texto, junto com todos os arquivos em edição para proceder com a recompilação e testes. Em uma sessão de desenvolvimento, o tempo gasto seria múltiplas vezes superiores ao tempo gasto com a utilização dos comandos de tarefa.

Para esse controle de tarefas são utilizados os comandos "bg" e "fg".

bg

bg é a abreviação para *background* , e a função principal deste comando é de colocar os programas para rodarem em segundo plano.

Sintaxe:

```
bg %identificação_trabalho
```

Exemplo:

```
bg %2
```

O exemplo acima coloca o trabalho de número 2 em segundo plano.

Ao executar um programa, pode ser desejável colocá-lo imediatamente em segundo plano. Isso é obtido através da seguinte sintaxe:

Sintaxe:

```
programa &
```

Exemplo:

```
$xterm &


[1] 1317
```

Nesse exemplo foi solicitada a abertura do **xterm** em *background*. Logo após a solicitação apareceram dois valores e por último foi devolvido o *prompt* de comando para o usuário.

Os valores apresentados são um entre colchetes e outro sem colchetes.

O valor entre colchetes representa o job (trabalho) ao qual esse programa está associado. Esse valor é auto-incrementável, e único em uma determinada sessão em um determinado console. Esse é o valor a ser utilizado em uma possível chamada ao comando "fg".

O segundo valor apresentado é a identificação do processo ao qual esse programa pertence.

O kill -9 pode enviar um sinal de parada do processo, e com isso terminar um programa. Com o segundo valor é possível parar o trabalho de processo em *background*. Mas atenção! O valor apresentado sem colchetes é o "pid" para o *background*, assim que este terminar de processar, morrerá automaticamente o processo e o programa solicitado poderá ter um valor diferente do apresentado.

fg

Para trazer de volta a interatividade ao nosso editor de texto do exemplo anterior, utilizaríamos o comando "fg", abreviação de *foreground* (visão superior, à frente). Note que um programa em segundo plano pode até continuar gerando mensagens na tela, mas não consegue receber entrada de dados do usuário.

O número de identificação da tarefa é usada como argumento no comando "fg". Para ver as tarefas existentes em sua sessão, lembre-se do comando jobs.

Sintaxe:

```
fg %x
```

onde *x* é o valor apresentado pelo comando jobs ou quando bg é acionado para trazer o comando para *background*.

ou

```
fg pid
```

onde *pid* é o número identificador do processo.

Exemplo:

```
$netscape &

[1] 1738

[aluno@lab20 aluno]$fg %1

netscape
```

ou

```
$netscape &

[1] 1738

$fg 1738

netscape
```

Em ambos os exemplos foi solicitado o netscape em background (com a utilização do "&"), com os valores [1] (jobs) e o 1738 (pid).

No exemplo acima, o programa netscape foi colocado para rodar em foreground com o seu número identificador de tarefa.

No segundo exemplo foi colocado para rodar com o pid dado para o background, lembrando-se que o pid apresentado ao colocar um programa para ser aberto em background poderá ser diferente do pid dado ao mesmo programa depois de aberto.

Prioridade de processos

Ao solicitar a abertura de um programa, é criado uma tabela sobre o processo gerado, ficando armazenado todos os dados do mesmo. Quando um programa é posto para processar, ele irá esperar, em uma fila, por sua hora de execução no processador, porém é possível colocar prioridade sobre este programa no tempo do processador. Ao modificar a prioridade de um processo ele é colocado em uma posição mais a frente na fila de processos sendo executado antes de outros com menor prioridade.

Os níveis de prioridade podem variar entre -19 até 20, sendo que o nível mais alto está em -19 variando sua importância até o nível mais baixo em 20. Os programas ao serem gerados têm inserido em sua tabela de processo o nível 10, até mesmo para o super usuário, isso porque todos

os usuários serão tratados iguais. Porém existe uma diferenciação quanto aos valores, pois os usuários comuns podem variar o nível de prioridade dentro dos valores positivos (até 0) enquanto que o superusuário poderá alcançar os valores negativos.

O comando que realiza o controle destes níveis são o nice e o renice. A diferença entre esses dois comandos está no momento da passagem do valor de prioridade. Enquanto o nice deverá passar o valor antes do processo ser inicializado, o renice pode alterar o valor do mesmo com o programa já em execução.

nice

Sintaxe para o nice:

```
nice [prioridade] comando
```

Prioridade:

-n - sendo que n é o valor da prioridade a ser passada junto com o comando a ser executado

renice

Sintaxe para o renice:

```
renice [prioridade] [-p pid] [-g grupo] [-u usuário]
```

Prioridade:

[+][-]n - sendo que n é o valor da prioridade a ser passada

O valor da prioridade do renice pode variar entre -19 e 20, sendo que quanto menor o valor maior será a sua prioridade.

Pid - é o valor identificador do processo

Grupo - utilize esta opção para alterar a prioridade de todos os processos executados por determinado grupo

Usuário - utilize esta opção para alterar a prioridade de todos os processos executados por determinado usuário

Relatório Individual:

1. **Qual a quantidade de memória RAM que possui o sistema, no momento, quanto desta memória está sendo usada e quanta memória está livre?**

2. **Se a memória usada pelos processos em execução ultrapassar a RAM disponível, a área de swap estará sendo usada. Assim, quanto swap está sendo usado?**

3. **Qual a utilidade de conhecer os detalhes sobre a alocação de memória do sistema de gerenciamento de memória?**

4. **Qual é o identificador do processo (PID) que identifica a tarefa que corresponde à execução do programa *bash*?**

5. **Utilizando o PID obtido na questão anterior informe o total de memória RAM, que o processo está usando, e a quantidade de memória swap.**

6. **Utilizando o comando top identificar o total de tarefas que está ativa, quantas estão em execução, quantas em dormência e quantas paradas.**

7. **Utilizando o comando *Kill* encerre a aplicação *bash*. Transcreva o comando executado.**

8. **Priorização de processos e tarefas em segundo plano, Execute e relacione as sintaxes dos comandos executados e os respectivos retornos:**

    a. **Execute o comando "top" com o nível de prioridade 15. O que acontece na linha que lista a tarefa?**

    b. **Coloque o comando "top" para execução em segundo plano;**

    c. **Altere o nível de prioridade da tarefa "top" para -10; O que acontece?**

    d. **Altere para o usuário administrado e repita os comandos (a,b e c). O que acontece?**

    e. **Liste as tarefas em segundo plano; O tarefa que executa o comando "top" aparece?**

f. **Coloque a tarefa “top” em primeiro plano. Indique qual o nível de prioridade que a tarefa “top” apresenta.**

Referencia Bibliográfica:

http://wiki.sintectus.com/bin/view/GrupoLinux/LicaoControlandoProcessos