Sistemas Operacionais de Redes

Tecnologia em Redes de Computadores

Aula 03

Prof. Me. Henrique Martins

Fatec Bauru

SISTEMAS OPERACIONAIS DE REDES

Prof. Me. Henrique Martins

Aula 03

- Montando partições
- Editor de Texto: VI
- Tar e gzip
- Controlando Processos

Montando partições

Ao colocar um CD ou um pendrive no computador o sistema precisa montar esse dispositivo.

- Essa ação é automática.

O comando para montar um dispositivo é mount

```
# mount /dev/sda /mnt/usb
# mount /dev/hdc /mnt/cdrom
```

Para desmontar utilize o comando umount

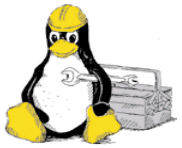
```
# umount /mnt/usb
# umount /mnt/cdrom
```

Montando partições

Crie um diretório em /mnt com o nome que desejar. Este diretório será onde iremos montar o pendrive.

 # mkdir /mnt/usb

Agora falta só montar:
 # mount -t vfat -o umask=0000 /dev/sda1 /mnt/usb
(-o de opções, no caso umask, onde a permisão de todos)
Se seu hd for sata provavelmente o comando será:
 # mount -t vfat -o umask=0000 /dev/sdb1 /mnt/usb



Pronto, seu pendrive já está montando:
 # cd /mnt/usb
 # ls

Montando partições

SUGESTÃO: Provavelmente você só vai poder montar o pendrive se for root, mas seria chato. Se toda vez que for montar o pendrive você tiver que logar como root, principalmente se você não for o root, a solução é:

Pedir ao root (se não for você) para adicionar a seguinte linha no */etc/fstab*:

/dev/sda1 /mnt/nome vfat noauto,user,umask=000 0 0

Explicando:

/dev/sda1: dispositivo onde está o pendrive;
/mnt/nome: diretório onde vai ser montado;
vfat: tipo do sistema de arquivos;
noauto: para não montar automaticamente ao iniciar (*importante*);
user: para qualquer usuário poder montar;
umask=000: permissão para todos escreverem, lerem e gravarem.

Editor de Texto: VI

MODO TEXTO
Subcomandos de inserção de texto:

i	insere texto antes do cursor
r	insere texto no início da linha onde se encontra o cursor
a	insere texto depois do cursor
A	insere texto no fim da linha onde se encontra o cursor
o	adiciona linha abaixo da linha corrente
O	adiciona linha acima da linha corrente
Ctrl + h	apaga último caracter
Ctrl + w	apaga última palavra minúscula
Esc	passa para o modo comando

Editor de Texto: VI

MODO COMANDO:
Subcomandos para Movimentação pelo Texto:

Ctrl+f	passa para a tela seguinte.
Ctrl+b	passa para a tela anterior.
H	move o cursor para a primeira linha da tela.
M	move o cursor para o meio da tela.
L	move o cursor para a última linha da tela.
H	move cursor para caracter a esquerda.
J	move cursor para linha abaixo.
K	move o cursor para linha acima.
l	move cursor para caracter a direita.
w	move cursor para início da próxima palavra (Ignora pontuação).
W	move cursor para início da próxima palavra (Não ignora pontuação).
b	move cursor para início da palavra anterior (Ignora pontuação).
B	move cursor para início da palavra anterior (Não ignora pontuação).
0(zero)	move cursor para início da linha corrente.
^	move cursor para o primeiro caracter não branco da linha.
$	move cursor para o fim da linha corrente.
nG	move para a linha n.
G	move para a última linha do arquivo.

Editor de Texto: VI

Subcomandos para Localização de Texto:

/palavra	procura pela palavra ou caracter acima ou abaixo do texto.
?palavra	move para a ocorrência anterior da palavra(para repetir a busca usar n).
n	repete o ultimo / ou ? comando.
N	repete o ultimo / ou ? comando na direção reversa.
Ctrl+g	mostra o nome do arquivo, o número da linha corrente e o total de linhas.

Editor de Texto: VI

Subcomandos para Alteração de Texto:

x	deleta um caracter que esta sobre o cursor.
dw	deleta a palavra, do inicio da posicao do cursor ate o fim.
dd	deleta a linha inteira onde o cursor estiver.
D	deleta a linha a partir da posicao do cursor em diante.
rx	substitui o caracter sob o cursor pelo especificado x (é opcional indicar o caracter).
Rtexto	substitui o texto corrente pelo texto indicado (opcional indicar o texto adicionado).
cw	substitui a palavra corrente. Pode-se inserir o novo conteudo da palavra automaticamente.
cc	substitui a linha corrente. Pode-se inserir o novo conteúdo da linha automaticamente.
C	substitui restante da linha corrente. Pode-se inserir o texto logo após o comando.
u	desfaz a última modificação.
U	desfaz todas as modificações feitas na linha (se o cursor não mudou de linha).
J	une a linha corrente a próxima.
s:/velho/novo	substitui a primeira ocorrêndcia de "velho" por "novo".

Editor de Texto: VI

Subcomandos para Salvar o Texto:

:x	salvar as mudanças feitas no arquivo e sai do editor.
:wq	salvar as mudanças feitas no arquivo e sai do editor.
:w < nome-arq >	salva o arquivo corrente com o nome especificado. Continua edição normalmente.
:w! < nome-arq >	salva (de modo forçado) o arquivo corrente no arquivo especificado.
:q	sai do editor. Se mudanças não foram salvas é apresentada mensagem de advertência.
:q!	sai do editor sem salvar as mudanças realizadas.

Alguns Comando Linux

date: mostra data e hora;

ps: relata os processos em execução;

kill: encerra um ou mais processos em andamento;

history: mostra os comandos que o usuário já digitou;

lpr: imprime um arquivo (exemplo: lpr arquivo);

lpq: mostra o status da fila de impressão;

lprm: remove trabalhos da fila de impressão;

Alguns Comando Linux

history: O comando history exibe os últimos comando digitados no bash.

history : Exibir listagem dos últimos comandos

history –c : Remover os últimos comandos

vi .bash_history : Verificar onde são armazenado os comandos

history -c : Limpar

history -r : Recuperar

Tar e gzip

- O **Tar** e o **gzip** são duas ferramentas utilizadas em sistemas operacionais baseados no Unix, como o GNU/Linux, para o "empacotamento" e para a compressão de arquivos, respectivamente.

- Embora seja perfeitamente possível usar qualquer desses programas de forma individual, a utilização de ambos ao mesmo tempo é muito comum e útil.

Tar

- **Tar**, sigla de ***T**ape **Ar**chive*. O Tar é muito simples de entender: ele "empacota" vários arquivos em um só, isto é, faz com que um único arquivo contenha vários outros. Assim, é possível, por exemplo, armazenar em único arquivo as cópias de documentos existentes na pasta de um usuário.

tar [parâmetros] [nome_do_arquivo_tar] [arquivos_de_origem]

-c - cria um novo arquivo tar;
-t - exibe o conteúdo de um arquivo tar;
-p - mantém as permissões originais do(s) arquivo(s);
-r - adiciona arquivos a um arquivo tar existente;
-f - permite especificar o arquivo tar a ser utilizado;
-v - exibe detalhes da operação;
-w - pede confirmação antes de cada ação no comando;
-x - extrai arquivos de um arquivo tar existente;
-z - comprime o arquivo tar resultante com o gzip;
-C - especifica o diretório dos arquivos a serem armazenados (note que, neste caso, a letra é maiúscula).

Tar

- O comando abaixo cria o arquivo **fatec.tar**, que contém os arquivos **laboratorio.txt** e **sala.txt**. Aqui, você deve ter reparado que é possível combinar parâmetros. Neste exemplo, isso ocorreu com **-c** e **-f**.

```
tar –cf fatec.tar laboratorio.txt sala.txt
```

- No exemplo abaixo, o diretório **aula5** tem todo o seu conteúdo compactado no arquivo **aula5.tar**, só que os detalhes são exibidos graças à opção **-v**:

```
tar –cvf aula5.tar aula5
tar –cvf /home/fatec/aula5.tar /home/fatec/aula5
```

Tar

- O exemplo a seguir lista o conteúdo do arquivo **redes.tar**:

```
tar -tf redes.tar
```

- Por sua vez, o comando abaixo faz com que todos os arquivos de **redes.tar** sejam extraídos (neste ponto, você certamente já sabe as funções dos parâmetros **x**, **v** e **f** no comando):

```
tar -xvf redes.tar
```

- Já no comando a seguir, apenas o arquivo **sala.txt** é extraído:

```
tar -xvf redes.tar sala.txt
```

Exercício

- Na pasta /home/USUARIO criar um diretório chamado **aula3**

- Dentro do diretório aula3 criar 3 arquivos ".txt"
  - Laboratorio.txt
  - Rede.txt
  - Sala3.txt
- Em seguida empacotar os 3 arquivos em um arquivo chamado **arquivos.tar (só que deve ser empacotado pelo método por arquivos)**
- E em seguida empacotar o diretório **aula3** para um arquivo chamado **aula3.tar (só que deve ser empacotado pelo método por diretório)**

gzip

- A ferramenta Tar, por si somente, serve apenas para juntar vários arquivos em um só. No entanto, o programa não é capaz de diminuir o tamanho do arquivo resultante, isto é, de compactá-lo. É neste ponto que entra em cena o **gzip** (*GNU zip*) ou outro compactador de sua preferência.
- Se utilizado isoladamente, o gzip faz uso da seguinte sintaxe:

```
gzip [parâmetros] [nome_do_arquivo]
    -c - extrai um arquivo para a saída padrão;
    -d - descompacta um arquivo comprimido;
    -l - lista o conteúdo de um arquivo compactado;
    -v - exibe detalhes sobre o procedimento;
    -r - compacta pastas;
    -t testa a integridade de um arquivo compactado.
```

- Ainda no que se refere às opções de parâmetros, é possível utilizar uma numeração de 1 a 9 para indicar o nível de compactação. Quanto maior o número, maior será a compactação do arquivo.

gzip

```
gzip fatec.odt
```

- O comando acima compacta o arquivo **fatec.odt**. Note que os arquivos compactados com gzip recebem a extensão .gz.

```
gzip -d fatec.odt.gz
```

- O comando acima descompacta o arquivo **fatec.odt.gz**.

```
gzip -1 fatec.ods
```

- O procedimento acima faz com que o arquivo **fatec.ods** seja compactado considerando o nível mais baixo de compreensão.

Usando Tar e gzip

- Os comandos Tar e gzip podem ser utilizados juntos. Muitas vezes, é necessário juntar arquivos e, ao mesmo, fazer com que o arquivo resultante, além de conter todos os outros, também seja compactado. É aí que entra em cena a capacidade de juntar arquivos do Tar com a capacidade de compactação do gzip.
- Para utilizar ambos ao mesmo tempo, o procedimento é muito simples: basta aplicar o comando **tar** com o parâmetro **-z**. O arquivo resultante desse procedimento receberá a extensão .tar.gz.
- Neste ponto, vemos um comando bastante usado na instalação de programas e bibliotecas:

```
tar -zcvf nome_do_arquivo.tar.gz
```

Usando Tar e gzip

`tar -zcvf nome_do_arquivo.tar.gz`

- a letra **z** deve ser usada porque o arquivo foi compactado com gzip;
- a letra **c** grava o resultado na saída padrão e mantém o arquivo original inalterado.;
- a letra **v** exibe os detalhes do procedimento;
- a letra **f** especifica qual arquivo será usado nesta atividade.

- Suponha, agora, que você queira deixar em um único pacote os arquivos **fatec.png**, **laboratorio.txt** e **sala.odt**. O arquivo resultante terá o nome **redes.tar.gz**. Eis o comando que utilizaremos para este exemplo:

```
tar -zcvf redes.tar.gz fatec.png laboratorio.txt sala.odt
```

Usando Tar e gzip

- Note que o comando é muito parecido com o procedimento de descompactação do exemplo anterior, com a diferença de que o parâmetro **c** foi utilizado no lugar de **x**, pois o objetivo aqui é criar um arquivo novo, e não fazer a extração de um já existente.
- Para extrair o conteúdo desse arquivo, basta executar o comando abaixo:

```
tar -zcvf redes.tar.gz
```

- Se você quiser extrair apenas um dos arquivos contidos no arquivo compactado, basta indicá-lo no final do comando. Por exemplo, suponha que você queira extrair o arquivo **fatec.png** de **redes.tar.gz**. Eis o que você deve digitar:

```
tar -zcvf redes.tar.gz fatec.png
```

Exercícios

- No diretório /home criar uma pasta chamada “Quintafeira”, em seguida crie 3 ou mais arquivos no formato “.txt” dentro da pasta Quintafeira, pelo **VI ou NANO ou qualquer outro editor de texto**.

- Em seguida utilize os comandos **Tar** e **gzip**, para compactar a pasta Quintafeira

Inicialização e desligamento

- O Linux é um sistema operacional complexo, e ligar e desligar sistemas Linux é mais complicado do que simplesmente pressionar o botão de energia. Ambas as operações têm de ser realizadas corretamente caso se queira que o sistema permaneça saudável.
- Embora o processo de inicialização (*bootstrapping*) sempre tenha sido um pouco misterioso, tudo era mais simples nos dias em que os fabricantes controlavam cada aspecto do hardware e software do sistema. Agora que o Linux executa em hardware de PC, o procedimento de inicialização precisa seguir regras para PC e lidar com diferentes configurações.

Inicialização

- *Bootstrapping* é o termo usado em inglês para "inicializar um computador". Durante a inicialização, o kernel é carregado na memória e começa a ser executado. Uma série de tarefas de inicialização é executada e o sistema é então tornado disponível aos usuários.
- O momento da inicialização é um período de vulnerabilidade especial. Erros em arquivos de configuração, equipamentos faltantes ou não confiáveis, e sistemas de arquivos danificados podem impedir que um computador comece a funcionar.

Inicialização

- Quando um computador é ligado, ele executa o código de boot que se encontra armazenado em ROM. Esse código, por sua vez, tenta descobrir como carregar e iniciar o kernel. O kernel investiga o hardware do sistema e depois gera o processo **init** do sistema, que sempre é PID 1.
- PID é o número de identificação de processos
- O kernel atribui um número de identificação exclusivo a cada processo. A maioria das chamadas de sistemas e comandos que manipulam processos exige que especifiquemos um PID para identificar o objeto da operação. Os PIDs são atribuídos na ordem em que os processos são criados.

Inicialização

- Os sistemas Linux podem ser inicializados tanto em modo automático ou em modo manual.
- No modo automático o sistema executa o procedimento de inicialização completo por sua conta, sem qualquer ajuda externa.
- No modo manual, o sistema segue o procedimento automático até um determinado ponto, mas então passa o controle a um operador antes de a maioria dos scripts de inicialização ter sido executada.
- Na operação do dia-a-dia, a inicialização automática é quase exclusivamente utilizada.

Inicialização

- Um processo de inicialização do Linux típico consiste em seis etapas distintas.
  - Carregamento e inicialização do kernel
  - Detecção e configuração de dispositivos
  - Criação de threads do kernel para processos de sistemas espontâneos
  - Intervenção do operador (somente para inicialização manual)
  - Execução dos scripts de inicialização do sistema
  - Operação multiusuário
- Os administradores têm pouco controle sobre a maioria dessas etapas. Efetuamos a maior parte da configuração de inicialização editando os scripts de inicialização do sistema.

Reinicializando e Desligando

- Em sistemas operacionais voltados ao mercado de consumo, reinicializar o sistema operacional é um primeiro tratamento apropriado para quase todos os problemas. Em um sistema Linux, é melhor pensar duas vezes antes de reiniciar o sistema.
- Os problemas no Linux tendem a ser mais sutis e mais complexos; portanto, reinicializar cegamente é eficiente apenas em uma pequena porcentagem de casos. Os sistemas Linux também levam um longo tempo para inicializar e vários usuários poderão ser incomodados.

Reinicializando e Desligando

- Diferentemente da inicialização, que pode ser feita essencialmente de uma única maneira, o desligamento ou reinicialização pode ser feito de várias maneiras:
  - Desligando a energia
  - Utilizando o comando **shutdown**
  - Utilizando os comando **halt** e **reboot**
  - Utilizando **telnit** para alterar os níveis de execução de **init**
  - Utilizando o comando **poweroff** para informar o sistema para desligar a energia.

Controlando processos

- Um processo é uma abstração utilizada pelo Linux para representar um programa em execução. É o objeto por meio do qual o uso de memória, tempo de processador e recursos de E/S de um programa podem ser gerenciados e monitorados.
- Um processo é constituído de um espaço de endereço e um conjunto de estruturas de dados dentro do kernel. O espaço de endereço é um conjunto de páginas da memória que o kernel marcou para ser empregado pelo processo.

Controlando processos

- As estruturas de dados internas do kernel registram vários tipos de informações sobre cada processo. Algumas das mais importantes são:
  - O mapa de espaço de endereço do processo
  - O status atual do processo (espera, parado, em execução, etc)
  - A prioridade de execução do processo
  - Informações sobre o recurso que o processo utilizou
  - Informações sobre s arquivos e as portas de rede que o processo abriu
  - A mascara de sinalização do processo (um registro de quais sinais são bloqueados)
  - O proprietário do processo

PID: número de identificação de processo

- O kernel atribui um número de identificação exclusivo a cada processo. A maioria das chamadas de sistemas e comandos que manipulam processos exige que especifiquemos um PID para identificar o objeto da operação.
- Os PIDs são atribuídos na ordem em que os processos são criados.

PPID: PID pai

- O Linus não fornece uma chamada de sistema que crie um processo executando um determinado programa. Em vez disso, um processo existente tem de ser clonar para criar um processo novo. O clone pode então trocar o programa que ele está executando por um outro diferente.
- Quando um processo é clonado, o original é chamado de pai e, a cópia, de filho. O atributo PPID de um processo é o PID do pai a partir do qual ele foi clonado.

PS: Monitorando processos

- **ps** é a principal ferramenta do administrador de sistemas para monitoramento de processos. Você pode utilizá-lo para exibir o PID, a prioridade, e o terminal de controle de processos.
- Ele também dá informações sobre quanta memória foi consumida e seu estado atual (em execução, parado, dormente, etc).
- Para obter uma visão geral dos processos em execução no sistema execute o comando **ps aux**.

```
debian:/etc/apt# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.9  0.0   2100   684 ?        Ss   17:12   0:02 init [2]
root         2  0.0  0.0      0     0 ?        S<   17:12   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   17:12   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S<   17:12   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   17:12   0:00 [watchdog/0]
root         6  0.0  0.0      0     0 ?        S<   17:12   0:00 [events/0]
root         7  0.0  0.0      0     0 ?        S<   17:12   0:00 [khelper]
root        39  0.0  0.0      0     0 ?        S<   17:12   0:00 [kblockd/0]
root        41  0.0  0.0      0     0 ?        S<   17:12   0:00 [kacpid]
root        42  0.0  0.0      0     0 ?        S<   17:12   0:00 [kacpi_notify]
root       103  0.0  0.0      0     0 ?        S<   17:12   0:00 [kseriod]
root       139  0.0  0.0      0     0 ?        S    17:12   0:00 [pdflush]
root       140  0.0  0.0      0     0 ?        S    17:12   0:00 [pdflush]
root       141  0.0  0.0      0     0 ?        S<   17:12   0:00 [kswapd0]
root       142  0.0  0.0      0     0 ?        S<   17:12   0:00 [aio/0]
root       588  0.0  0.0      0     0 ?        S<   17:12   0:00 [ksuspend_usbd]
root       589  0.0  0.0      0     0 ?        S<   17:12   0:00 [khubd]
root       596  0.0  0.0      0     0 ?        S<   17:12   0:00 [scsi_eh_0]
root       691  0.0  0.0      0     0 ?        S<   17:12   0:00 [ata/0]
root       692  0.0  0.0      0     0 ?        S<   17:12   0:00 [ata_aux]
root       756  0.0  0.0      0     0 ?        S<   17:12   0:00 [kjournald]
root       832  0.2  0.1   3080  1676 ?        S<s  17:12   0:00 udevd --daemon
root      1210  0.0  0.0      0     0 ?        S<   17:12   0:00 [kpsmoused]
root      1223  0.0  0.0      0     0 ?        S<   17:12   0:00 [kgameportd]
daemon    1628  0.0  0.0   1892   508 ?        Ss   17:13   0:00 /sbin/portmap
statd     1640  0.0  0.0   1956   724 ?        Ss   17:13   0:00 /sbin/rpc.statd
root      1881  0.0  0.0   2180   388 ?        S<s  17:13   0:00 dhclient3 -pf /
```

PS: Monitorando processos

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

USER – Nome de usuário do proprietário do processo

PID – ID do processo

%CPU – Porcentagem dos recursos de CPU que esse processo está utilizando

%MEM – Porcentagem da memória real que esse processo esta utilizando

VSZ – Tamanho virtual do processo

RSS – Tamanho configurado residente (número de páginas na memória)

TTY – ID do terminal controlador

STAT – Status do processo atual

START – A data/hora em que o processo foi iniciado

TIME – O tempo de CPU que o processo consumiu

COMMAND – Nome e argumento de comando

PS: Monitorando processos

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

STAT – Status do processo atual:

R = Executável

D = Em repouso não interrompível

S = Em repouso (<20 sec)

T = Rastreado ou parado

Z = Zumbi

Flags adicionais:

W = Processo sofre swap

< = O processo tem prioridade mais alta que a normal

N = O processo tem prioridade mais baixa que a normal

L = Algumas páginas são bloqueadas no núcleo

s = O processo é um lider de sessão

Estado de processos

- Os processos podem ser:
- **Executável**: é aquele que está pronto para ser executado toda vez que houver tempo de CPU.
- **Dormente**: estão aguardando a ocorrência de um evento específico.
- **Zumbi**: são processos que terminaram a execução, mas ainda não tiveram seus status coletados.
- **Parado**: Ser parado é similar a adormecer, mas não há como sair do estado parado a não ser ter algum outro processo que nos acorda.

TOP: Monitorando os processos

- Uma vez que comandos como ps oferecem apenas um instantâneo de seu sistema em um dado momento, normalmente é difícil ter visão geral do que realmente está acontecendo.
- O comando top fornece um resumo regularmente atualizado dos processos ativos e o uso de recursos

 # top –d 10

- Inicia o top e atualiza-o a cada 10 segundos
- Para encerrar a execução do comando top, pressione <CTRL>+<C>, ou **q**.

Fatec
Bauru

SISTEMAS OPERACIONAIS DE REDES
Prof. Me. Henrique Martins

```
top - 17:39:19 up 26 min,  2 users,  load average: 0.00, 0.00, 0.04
Tasks: 117 total,   2 running, 115 sleeping,   0 stopped,   0 zombie
Cpu(s):  7.7%us,  8.1%sy,  0.0%ni, 84.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1019836k total,   289660k used,   730176k free,    31912k buffers
Swap:   409616k total,        0k used,   409616k free,   133628k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2567 root      20   0 36404  11m 6316 S 10.0  1.1   0:18.20 Xorg
 2680 henrique  20   0 16088 5848 4732 S  3.3  0.6   0:00.72 gnome-screensav
 2683 henrique  20   0 36472  18m  11m S  2.7  1.9   0:01.56 gnome-panel
 2371 root      20   0  6768 1124  732 S  0.7  0.1   0:00.12 kerneloops
 2869 root      20   0 37564  14m 9556 R  0.7  1.4   0:02.06 gnome-terminal
 3100 root      20   0  2392 1140  884 R  0.7  0.1   0:00.08 top
    1 root      20   0  2100  684  588 S  0.0  0.1   0:02.32 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 kthreadd
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:00.12 ksoftirqd/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:00.16 events/0
    7 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 khelper
   39 root      15  -5     0    0    0 S  0.0  0.0   0:00.18 kblockd/0
   41 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   42 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  103 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  139 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  140 root      20   0     0    0    0 S  0.0  0.0   0:00.06 pdflush
  141 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kswapd0
  142 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
  588 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ksuspend_usbd
```

top: monitorando processos

- **top -opção**
- Entre as opções, tem-se as que se seguem:

-d - atualiza o top após um determinado período de tempo (em segundos). Para isso, informe a quantidade de segundos após a letra d. Por exemplo: top -d 30;
-c - exibe a linha de comando ao invés do nome do processo;
-i - faz o top ignorar processos em estado zumbi;
-s - executa o top em modo seguro.

jobs: monitorando processos

- **jobs -** serve para visualizar os processos que estão parados ou executando em segundo plano (background). Quando um processo está nessa condição, significa sua execução é feita pelo kernel sem que esteja vinculada a um terminal. Em outras palavras, um processo em segundo plano é aquele que é executado enquanto o usuário faz outra coisa no sistema.
- Uma dica para saber se o processo está em background é verificar a existência do caractere & no final da linha. Se o processo estiver parado, geralmente a palavra "stopped" aparece na linha, do contrário, a palavra "running" é exibida.

jobs: monitorando processos

- A sintaxe do jobs é:

jobs -opção

- As opções disponíveis são:

-l - lista os processos através do PID;
-r - lista apenas os processos em execução;
-s - lista apenas os processos parados.

- Se na linha de um processo aparecer o sinal positivo (+), significa que este é o processo mais recente a ser paralisado ou a estar em segundo plano. Se o sinal for negativo (-), o processo foi o penúltimo. Note também que no início da linha um número é mostrado entre colchetes. Muitos confundem esse valor com o PID do processo, mas, na verdade, trata-se do número de ordem usado pelo jobs.

fg e bg: monitorando processos

- **fg e bg:** o fg é um comando que permite a um processo em segundo plano (ou parado) passar para o primeiro (foreground), enquanto que o bg passa um processo do primeiro plano para o segundo. Para usar o bg, deve-se paralisar o processo. Isso pode ser feito pressionando-se as teclas Ctrl + Z no teclado. Em seguida, digita-se o comando da seguinte forma:

bg +número

- O número mencionado corresponde ao valor de ordem informado no início da linha quando o comando jobs é usado.
- Quanto ao comando fg, a sintaxe é a mesma:

fg +número

pstree: monitorando processos

- **pstree:** esse comando mostra processos relacionados em formato de árvore. Sua sintaxe é:

pstree -opção PID

- Entre as opções, tem-se:

-u - mostra o proprietário do processo;
-p - exibe o PID após o nome do processo;
-c - mostra a relação de processos ativos;
-G - usa determinados caracteres para exibir o resultado em um formato gráfico.

Kill

- O comando kill termina um processo em andamento. Um processo poderia ser, mais comumente, um programa. Um processo é identificado pelo PID (Process Identification Number)

 # kill -l (lista os nomes de sinais)

 # kill 1706 (apaga o processo, que neste exemplo, era somente um usuário que estava logado)

 # ps aux | grep squid (lista o processo que o squid está rodando. Com isso era necessário somente executar um kill nos processos do squid paara finalizálo)

 # kill -9 2991 (força a parada(mata) dos processo especificado)