UNIX: Gestão de processos

Programas e processos

Programas são arquivos em disco contendo instruções para execução pelo processador, enquanto **processos** são as execuções em andamento. Cada processo executando no sistema em um determinado momento é identificado por um número único, o PID - *Process IDentifier*. Além disso, cada processo possui outras informações que o caracterizam, como:

- Usuário proprietário (aquele que lançou o processo)
- Sessão de shell de onde foi lançado (se foi lançado através de um shell)
- Estado atual (*Running*, *Suspended*, *SWapped*, ...)
- Linha de comando usada para lançá-lo.
- Uso de memória e CPU
- etc.

No UNIX, os seguintes comandos permitem observar a atividade dos processos no sistema:

- `ps` : permite listar todos os processos ativos no sistema.
- `pstree` : mostra as dependências entre processos.
- `top` : mostra a atividade do sistema, atualizando a tela a cada N segundos. Permite interagir com os processos do usuário que está executando o comando (tecle "?" para uma tela de ajuda).
- `free` : apresenta informações gerais sobre uso de CPU e memória.
- `vmstat` : gera estatísticas sobre o uso de memória virtual, CPU, etc.
- `renice` : permite alterar a prioridade de base dos processos, que vai de -20 (máxima) a +20 (mínima), tendo como prioridade default o valor zero (0). Usuários normais só podem alterar as prioridades dos seus próprios processos, e só podem diminuir as prioridades.

Os comandos ps e pstree

Podemos visualizar os processos em execução no sistema através do comando `ps`, cuja execução sem parâmetros gera uma listagem como a seguinte:

```
$ ps
PID    TTY  STAT   TIME   COMMAND
14726  p3   S      0:02   -tcsh
17884  p3   R      0:00   ps
```

O comando `ps` aceita uma série de parâmetros, entre os quais os mais importantes são:

- `a` : mostra processos de outros usuários também (*all*).
- `u` : mostra listagem mais detalhada dos processos, com uso de memória e CPU
- `x` : mostra processos não conectados a terminais.
- `w` : mostra mais detalhes sobre as linhas de comando dos processos.

Para obter uma listagem completa dos processos em execução no sistema usam-se as opções **auxw**, que geram uma listagem como a que segue:

```
$ ps auxw
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
bin        315  0.0  0.1   780   312  ?  S   Apr 22   1:01 portmap
daemon     293  0.0  0.1   796   344  ?  S   Apr 22   0:00 /usr/sbin/atd
```

```
daemon   14477  0.0  0.1   796   396  ?  S    16:55   0:00 /usr/sbin/atd
edouard  15281  0.0  0.3  1504   908  ?  S    17:21   0:00 imapd
ftp      18017  0.1  0.3  1364   808  ?  S    18:31   0:00 ftpd: 200.134.48.57:
anonymous/gzbaron@ufpr.br: IDLE
gzbaron  20688  0.0  1.2  3980  3092  ?  S   Apr 27   0:00 wish ./tik.tcl
jamhour  14478  0.0  0.2  1196   660  ?  S N  16:55   0:00 sh
jamhour  14479  0.0  0.2  1024   684  ?  S N  16:55   0:00 wget -c
http://www.apache.org/dist/apache_1_3_6_win32.exe
jamhour  15188  0.0  0.4  1884  1200  p0 S    17:18   0:00 -tcsh
maziero  14726  0.0  1.0  3268  2580  p3 S    17:03   0:02 -tcsh
maziero  18019  0.0  0.1   860   496  p3 R    18:31   0:00 ps auxw
nobody     473  0.0  4.9 13972 12788  ?  S   Apr 22   2:30 squid
...
```

Os principais campos dessa listagem são:

- USER : o proprietário do processo, que pode ser quem o lançou ou, no caso de executáveis com o bit SUID habilitado, o proprietário do arquivo executável.
- PID : número do processo.
- %CPU : porcentagem da CPU usada pelo processo.
- %MEM : porcentagem da memória usada pelo processo.
- SIZE : memória total usada pelo processo.
- RSS : memória física (RAM) usada pelo processo.
- TTY : terminal ao qual o processo está ligado.
- STAT : status do processo (rodando, suspenso, ...).
- START : data de lançamento do processo.
- TIME : tempo total de CPU usado pelo processo.
- COMMAND : comando usado para lançar o processo.

O comando pstree é útil por mostrar a hierarquia existente entre os processos ativos no sistema:

```
$ pstree
init-+-apmd
     |-atd
     |-crond
     |-gpm
     |-httpd---8*[httpd]
     |-inetd-+-2*[imapd]
     |       `-3*[in.ftpd]
     |-kflushd
     |-klogd
     |-kpiod
     |-kswapd
     |-lpd
     |-mdrecoveryd
     |-6*[mingetty]
     |-miniserv.pl
     |-named
     |-nfsd---lockd---rpciod
     |-7*[nfsd]
     |-nmbd---nmbd
     |-portmap
     |-powerd
     |-rpc.mountd
     |-rpc.rquotad
     |-rpc.statd
```

```
     |-rpc.yppasswdd
     |-safe_mysqld---mysqld---mysqld---mysqld
     |-sendmail---sendmail
     |-smbd---3*[smbd]
     |-squid---squid-+-5*[dnsserver]
     |               `-unlinkd
     |-sshd---sshd---tcsh---pstree
     |-syslogd
     |-tcsh---kvt---tcsh
     |-update
     |-xdm
     |-xfs
     |-xntpd
     |-6*[xterm---tcsh]
     |-ypbind---ypbind
     `-ypserv
```

O comando top

O comando `top` é uma versão iterativa do comando `ps`, atualizando a listagem de processos a cada *n* segundos, e ordenando-os por uso de CPU e memória. Ele é bastante útil para compreender o que está sendo processado pela máquina em um determinado instante. Eis uma tela típica do comando top:

```
8:07am  up 2 days, 19:05,  1 user,  load average: 0.00, 0.00, 0.00
97 processes: 96 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  7.3% user,  1.8% system,  0.0% nice,  1.9% idle
Mem:  128032K av, 123480K used,   4552K free,  37904K shrd,  22404K buff
Swap: 385548K av,   9856K used, 375692K free                 49576K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
 1871 maziero   15   0  1024 1024   820 R       0  4.7  0.7   0:00 top
    1 root       0   0   104   56    40 S       0  0.0  0.0   0:03 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:06 kflushd
    3 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kpiod
    4 root       0   0     0    0     0 SW      0  0.0  0.0   0:03 kswapd
    5 root     -20 -20     0    0     0 SW>     0  0.0  0.0   0:00 mdrecoveryd
  109 root       0   0    60    0     0 SW      0  0.0  0.0   0:00 apmd
  262 bin        0   0   236  220   160 S       0  0.0  0.1   0:02 portmap
  277 root       1   0  4372 4332   212 S       0  0.0  3.3   0:52 ypserv
  292 root       0   0    76    0     0 SW      0  0.0  0.0   0:00 ypbind
  298 root       0   0   180  124    80 S       0  0.0  0.0   0:00 ypbind
  338 root       0   0    80   28    16 S       0  0.0  0.0   0:00 powerd
  386 daemon     0   0   136  104    64 S       0  0.0  0.0   0:00 atd
  400 root       0   0   164  112    76 S       0  0.0  0.0   0:00 crond
  414 root       0   0   144   80    40 S       0  0.0  0.0   0:04 inetd
  428 root       0   0  2004 1464   500 S       0  0.0  1.1   0:23 named
  435 root       0   0   392  336   264 S       0  0.0  0.2   0:11 sshd
```

Existem diversas interfaces gráficas para visualização dos processos do sistema, entre elas o `ktop`.

Background e foreground

O shell pemite lançar processos a partir da linha de comando. Normalmente cada comando digitado é executado na forma de um processo, e o shell espera o término do processo lançado para devolver o controle ao usuário. Essa forma de operação é chamada *foreground operation*, e torna o shell inacessível até que o processo lançado seja terminado.

Uma forma alternativa de lançamento de processos pelo shell é a chamada *background operation*, que consiste em lançar processos sem perder o controle do shell. Para isso bastar adicionar o caractere & ao final da linha do shell. Dessa forma, o processo será lançado em *background*, e o shell permanecerá disponível ao usuário. Por exemplo, para lançar o editor GEdit sem perder o controle do shell bastar digitar:

```
$ gedit &
```

As seguintes operações no shell são possíveis durante uma execução de processo em *foreground*:

- ^C : interrompe (mata) o processo, encerrando sua execução.
- ^Z : suspende o processo. um processo suspenso pode ser retornado à operação em *foreground*, através do comando `fg` (ou `fg %job`, quando houverem vários processos suspensos), ou transferido para operação em *background*, através do comando `bg` (ou `bg %job`, idem).

O comando `jobs` permite verificar quais os processos em *background* lançados pelo shell corrente. É importante que o shell só pode terminar (via comando `exit` ou `logout`) quando todos os seus jobs tiverem terminado.

o comando kill

O comando `kill` *-sinal* permite enviar sinais aos processos em execução. Somente o proprietário do processo e o usuário root podem enviar sinais. Os sinais mais usados são os seguintes:

- 15 (TERM): solicita ao processo seu término.
- 9 (KILL): mata o processo, sem mais delongas.
- 1 (HUP): solicita releitura dos arquivos de configuração (para os principais serviços básicos).

Assim, para eliminar um processo que não responde mais a comandos e recusa-se a terminar, basta executar `kill -9 pid`, onde *pid* corresponde ao número identificador do processo, obtido através dos comandos `ps` ou `top`.

Os comandos at e batch

O comando `at` permite agendar a execução de um comando para uma hora e data definida pelo usuário. Os resultados (via *stdout*) das execuções agendadas serão enviados ao usuário por e-mail. Vejamos um exemplo:

```
$ at 23:50 12/31/2031
> echo "Feliz 2032 a todos" | mail user@ufpr.br
> who
> ^D
```

O comando `batch` permite agendar uma execução de comandos sem data marcada, assim que a carga do sistema o permitir. Da mesma forma que no comando `at`, eventuais resultados (*stdout*) das execuções são enviados ao usuário por e-mail.

Verifique as páginas de manual dos comandos `at` e `batch` para maiores detalhes.

Agendando programas via crontab

Enquanto o comando `at` permite agendar uma tarefa para uma determinada data, o sistema `crontab` permite o agendamento de tarefas repetitivas ao longo do dia, semana, mês ou ano. Toda a informação sobre tarefas agendadas via `crontab` é mantida em um arquivo dentro do diretório `/var/spool/cron`, que não é diretamente acessível. O acesso ao arquivo é feito pelo comando `crontab`:

- `crontab -e` : editar o arquivo com as definições de tarefas
- `crontab -l` : listar o conteúdo do arquivo atual
- `crontab -r` : remover o arquivo.

A estrutura do arquivo de crontab é relativamente complexa mas permite muita flexibilidade. Vejamos um exemplo:

```
# Usar o shell /bin/sh to run commands
SHELL=/bin/sh

# enviar stdout para maziero, mesmo sendo o crontab de outro usuario
MAILTO=maziero

# Explicação do formato
# minuto, hora, dia, mês, dia da semana, comando
5 0 * * *       $HOME/bin/daily.job 1> $HOME/tmp/out 2>&1

# Rodar às 14:15 todo início de mês
15 14 1 * *     $HOME/bin/monthly.job

# Rodar às 22:30 todos os dias úteis
30 22 * * 1-5   mail -s "São 22:30, vá para casa !" maziero > /dev/null

# Outros exemplos
23 0-23/2 * * * echo "run 23 minutes after midn, 2am, 4am ..., everyday"
5 4 * * sun     echo "run at 04:05 every sunday"
```

Para maiores informações podem ser obtidas via páginas de manual (`man crontab` para o comando e `man 5 crontab` para o arquivo de configuração).

Exercícios

1. Execute o comando top em uma janela, mostrando apenas seus processos, enquanto faz os exercícios em outra janela.
2. Quantos processos você está executando neste momento?
3. O que faz o comando `kill -9 -1`?
4. Quais os programas com maior utilização de CPU que estão rodando?
5. Mostre a quantidade de processadores que seu computador tem conforme o que consta no arquivo `/proc/cpuinfo`.
6. Mostre a quantidade de memória disponível no seu computador conforme o que consta no arquivo `/proc/meminfo`.
7. Verifique quais os processos em atividade no sistema atualmente, identificando o uso de memória e CPU de cada um (dica: use o comando `ps auxw | more`). Identifique o significado de cada uma das colunas da listagem obtida (ver a página de manual). Quais os processos que mais consomem recursos do sistema ?
8. Efetue uma conexão SSH (Secure Shell) em um servidor. Do lado do cliente e do servidor, identifique os

processos envolvidos no estabelecimento da conexão SSH e como eles se relacionam.
9. Você é o administrador de um sistema e precisa desconectar imediatamente todos os usuários contectados via SSH. Como você poderia fazer isso?
10. Agende uma operação remoção dos arquivos `.bak` do seu diretório HOME e sub-diretórios para as 23:55, todas as segundas, quartas e sextas-feiras.