Laboratório de Redes de Computadores e Sistemas Operacionais

Controlando Processos

Fabricio Breve

Noções Básicas

- **Processo**: abstração usada pelo Linux para representar um programa em execução
  - Objeto através do qual podem ser gerenciados:
    - Uso de memória
    - Tempo do processador
    - Recursos de entrada e saída

Componentes de um Processo

- Mapa de espaços de endereço do processo
  - conjunto de páginas de memória que o kernel marcou para serem usadas pelo processo
    - Uma página de memória em um PC tem tipicamente 4 KB
  - Estado atual do processo (espera, parado, em execução, etc.)
  - Prioridade de execução do processo

Componentes de um Processo

- Informações sobre os recursos que o processo usou
- Máscara de sinalização do processo (um registro de quais sinais são bloqueados)
- Proprietário do processo

PID: número de identificação do processo

- O Linux não fornece uma chamada de sistema para criar um novo processo
- Um processo existente tem que se clonar para criar um processo novo, o clone então troca o programa que está executando
- Quando um processo se clona, o original é chamado de pai e a cópia de filho

UID e EUID

- UID: número de identificação do usuário da pessoa (cópia do EUID do processo pai)
  - Só o criador do processo e o root tem permissão para manipulá-lo
- EUID: UID “efetivo”, usado para determinar quais recursos e arquivos um processo pode executar.
  - UID e EUID normalmente são os mesmos

GID e EGID

- GID: número de identificação do grupo de um processo
- EGID: GID “efetivo”
- Um processo pode ser membro de vários grupos ao mesmo tempo

Escalonamento

- Determina quanto tempo de CPU um determinado processo recebe
- O kernel usa um algoritmo dinâmico para calcular prioridades, levando em conta:
  - A quantidade de tempo de CPU que o processo consumiu recentemente
  - O tempo que ele ficou aguardando para ser executado

Terminal de Controle

- A maioria dos processo está associada a um terminal de controle (tty)
  - Determina terminal de entrada/saída/erro padrão
  - Ao iniciar um comando no shell seu terminal se torna o terminal de controle do processo

Clico de Vida de um Processo

- Um processo faz uma cópia de si mesmo para criar um novo processo, usando a chamada de sistema *fork*
- Fork retorna o PID do filho recém-criado para o pai e 0 para o filho
- O processo filho assume outro papel
- Quando o sistema é inicializado, o kernel cria e instala vários processos, dentre eles o *init,* que é responsável pela maioria dos scripts de inicialização
- Quando um processo é completado, *init* chama uma rotina *_exit* para notificar o kernel de que ele está pronto para expirar
- O processo pai reconhece faz uma chamada *wait* para reconhecer a expiração do processo filho
- Filhos órfãos passam a ser filhos de *init*, que se encarrega de fazer a chamada *wait*

Sinais

- Podem ser enviados entre processos como meio de comunicação
- Podem ser enviados pelo driver de terminal para extinguir, interromper ou suspender processos quando teclas especiais foram pressionadas (Ctrl+C, Ctrl+Z)
- Podem ser enviados pelo administrador (via *kill*)
- Podem ser enviados pelo kernel quando um processo comete uma infração (ex.: divisão por zero)

Sinais

- Quando um sinal é recebido:
  - Se houver uma rotina de manipulação pra esse sinal em particular ela será usada
  - Caso contrário o kernel toma uma atitude padrão em nome do processo
- Os programas podem ignorar ou bloquear a chegada de sinais
  - Sinais ignorados são descartados
  - Sinais bloqueados é jogado em uma fila

Sinais

Sinais que todo administrador de sistemas deve conhecer.

#	Nome	Descrição	Padrão	Pode capturar?	Pode bloquear?	core dump?
1	HUP	Suspender	Encerrar	Sim	Sim	Não
2	INT	Interromper	Encerrar	Sim	Sim	Não
3	QUIT	Abandonar	Encerrar	Sim	Sim	Sim
9	KILL	Destruir	Encerrar	Não	Não	Não
a	BUS	Erro de barramento	Encerrar	Sim	Sim	Sim
11	SEGV	Falha de segmentação	Encerrar	Sim	Sim	Sim
15	TERM	Encerramento por parte do software	Encerrar	Sim	Sim	Não
a	STOP	Parar	Parar	Não	Não	Não
a	TSTP	Parada de teclado	Parar	Sim	Sim	Não
a	CONT	Continuar após parada	Ignorar	Sim	Não	Não
a	WHINCH	Modificado por janela	Ignorar	Sim	Sim	Não
a	USR1	Definido pelo usuário	Encerrar	Sim	Sim	Não
a	USR2	Definido pelo usuário	Encerrar	Sim	Sim	Não

a. Varia conforme a arquitetura de hardware; veja **man 7 signal**.

Sinais

- KILL e STOP: não podem ser capturados, bloqueados ou ignorados
  - KILL destrói o processo
  - STOP suspende o processo até que um sinal CONT seja recebido
    - CONT pode ser capturado ou ignorado, mas não bloqueado
- TSPS é um STOP mais “soft” (solicitação de parada), é gerado quando pressionamos Ctrl+Z
  - Programas que recebem esse sinal normalmente limpam seus estados e enviam um STOP para eles mesmos
  - TSPS poderia ser ignorado para impedir que um programa fosse encerrado pelo teclado

Sinais

- KILL: não pode ser bloqueado e encerra um processo em nível de S.O. Um processo jamais “recebe” esse sinal
- INT: enviado pelo driver de terminal quando digitamos Ctrl+C, programas simples podem sair (caso capturem o sinal) ou permitir que sejam eliminados (caso não capturem o sinal)
- TERM: solicitação para terminar completamente a execução, o processo deve limpar seu estado e sair

Sinais

- HUP: (tem duas interpretações)
  - solicitação de reinicialização (utilizada por vários deamons)
  - Gerados pelo driver de terminal numa tentativa de limpar os processos agregados a um terminal (sessão concluída / conexão perdida)
- QUIT: similar a TERM, porém seu padrão gera um core dump caso não seja capturado
  - Poucos programas canibalizam esse sinal e o interpretam de outra forma

Kill

- KILL pode enviar qualquer sinal, porém tipicamente é usado para encerrar um processo, por padrão o sinal enviado é TERM
  - Kill pode ser usado por usuários em seus próprios processos ou pelo root em qualquer processo
  - Sintaxe:
    - kill [-*sinal*] *pid*
      - sinal: número ou nome do sinal a ser enviado
      - pid: número de identificação do processo (use -1 para todos os processos exceto init)
  - Kill sem especificar o sinal não garante que o processo será terminado, pois TERM pode ser ignorado, bloqueado ou capturado. O sinal 9 não pode ser capturado e “garante” que o processo será eliminado.

Estados de Processos

- Há basicamente quatro estados de execução de um processo:
  - **Executável:** O processo pode ser executado
    - Apenas esperando tempo de CPU para processar seus dados
  - **Dormente:** O processo está aguardando algum recurso
    - Aguardando uma entrada de teclado ou de rede, um dado de disco, etc.
  - **Zumbi:** O processo está tentando se destruir
    - Teminou sua execução mas ainda não teve seus dados coletados
  - **Parado:** O processo é suspenso (não há permissão para ser executado)
    - Proibido administrativamente de executar (c/ um STOP ou TSTP e são reiniciados com CONT)

Nice e Renice

- A “gentileza” de um processo é uma dica numérica para o kernel em relação a como o processo deve ser tratado em relação aos outros processos lutando por recursos da CPU
- O intervalo de valores de nice vai de -20 a +19
  - Um valor “nice” (gentil) alto significa baixa prioridade
  - Um valor “nice” baixo significa alta prioridade
- O proprietário de um processo pode aumentar o “nice”, mas não diminuí-lo
- Root pode configurar “nice” da maneira que quiser
  - Pode até colocar um valor tão baixo que outros processos não poderão executar

Prioridade

- O escalonador normalmente faz um bom trabalho no gerenciamento da CPU (cada vez mais rápida), tornando a configuração manual de prioridades desnecessária na maioria dos casos
- O gargalo normalmente é no sistema de I/O (Ex.: discos rígidos), onde o valor do nice não tem nenhuma influência

Nice e Renice

- O valor de nice pode ser configurado no momento da criação do processo com o comando nice:
  - nice –n 5 -/bin/tarefademorada
- Pode ser configurado com renice:
  - renice -5 8829
- Atenção: alguns shell (não bash) incluem um comando nice com sintaxe diferente do sistema

ps: Monitorando Processos

- ps é a principal ferramenta para monitorar processos, ela fornece:
  - PID
  - UID
  - Prioridade
  - Terminal de controle
  - Informação de quanta memória foi consumida
  - Estado atual
- ps tem muitas implementações diferentes no UNIX
  - O Linux dá suporte a maioria delas e usa uma variável de ambiente para configurar qual deve ser usada
- Utilize **`ps aux`** para obter uma visão geral dos processos em execução

ps aux

```
root      2668  0.0  1.2  9484 3256 ?        Ss   16:53   0:00 sendmail: accepti
smmsp     2676  0.0  1.0  6620 2588 ?        Ss   16:53   0:00 sendmail: Queue r
root      2687  0.0  0.2  3012  532 ?        Ss   16:53   0:00 gpm -m /dev/input
root      2697  0.0  0.4  5424 1152 ?        Ss   16:53   0:00 crond
xfs       2720  0.0  0.6  4528 1612 ?        Ss   16:53   0:00 xfs -droppriv -da
root      2730  0.0  0.2  1724  640 ?        SNs  16:53   0:00 anacron -s
root      2739  0.0  0.2  1924  744 ?        Ss   16:53   0:00 /usr/sbin/atd
dbus      2758  0.0  0.5 14004 1324 ?        Ssl  16:53   0:00 dbus-daemon-1 --s
root      2771  0.0  0.2  5440  540 ?        Ss   16:53   0:00 rhnsd --interval
root      2780  0.0  0.4  3588 1040 ?        Ss   16:53   0:00 cups-config-daemo
root      2791  0.1  1.6  6880 4140 ?        Ss   16:53   0:03 hald
root      2803  0.0  0.1  2408  404 tty3     Ss+  16:53   0:00 /sbin/mingetty tt
root      2807  0.0  0.1  3224  404 tty4     Ss+  16:53   0:00 /sbin/mingetty tt
root      2811  0.0  0.1  2824  404 tty5     Ss+  16:53   0:00 /sbin/mingetty tt
root      2812  0.0  0.1  2424  404 tty6     Ss+  16:53   0:00 /sbin/mingetty tt
root      2813  0.0  0.9 11348 2352 ?        Ss   16:53   0:00 /usr/bin/gdm-bina
root      3256  0.0  1.1 11984 2936 ?        S    16:53   0:00 /usr/bin/gdm-bina
root      3263  0.2  3.3 10712 8444 ?        S    16:53   0:06 /usr/X11R6/bin/X
gdm       3400  0.0  4.2 20892 10740 ?       Ss   16:53   0:01 /usr/bin/gdmgreet
root      3517  0.0  0.5  4628 1284 ?        Ss   17:09   0:00 login -- root
root      3603  0.0  0.5  6024 1380 tty2     Ss+  17:12   0:00 -bash
root      4121  0.1  0.5  2948 1488 ?        Ss   17:26   0:00 login -- fbreve
fbreve    4206  0.3  0.5  5520 1372 tty1     Ss   17:27   0:00 -bash
fbreve    4250  0.0  0.2  3204  744 tty1     R+   17:27   0:00 ps aux
[fbreve@localhost ~]$ _
```

Campo	Conteúdo
USER	Nome do usuário do proprietário do processo
PID	ID (identificador) do processo
%CPU	Porcentagem dos recursos de CPU que este processo está usando
%MEM	Porcentagem da memória real que este processo está usando
VSZ	Tamanho virtual do processo
RSS	Resident Set Size (número de páginas na memória)
TTY	ID (identificador) de terminal de controle
STAT	Estado de processo atual R = Executável D = Espera no disco (ou a curto prazo) S = Dormente (<20 seg) T = Rastreado ou interrompido Z = Zumbi Flags adicionais: W = Processo paginado em disco < = O processo tem uma prioridade maior do que a normal N = O processo tem uma prioridade menor do que a normal L = Algumas páginas são bloqueadas no núcleo (kernel)
START	Horário em que o processo foi iniciado
TIME	Tempo de CPU que o processo consumiu
COMMAND	Nome do comando e argumentos[a]

a. Os argumentos podem ser truncados; acrescente o argumento **ww** para evitar isto. Os programas são capazes de modificar esta informação, de modo que não é necessariamente uma representação acurada da verdadeira linha de comando.

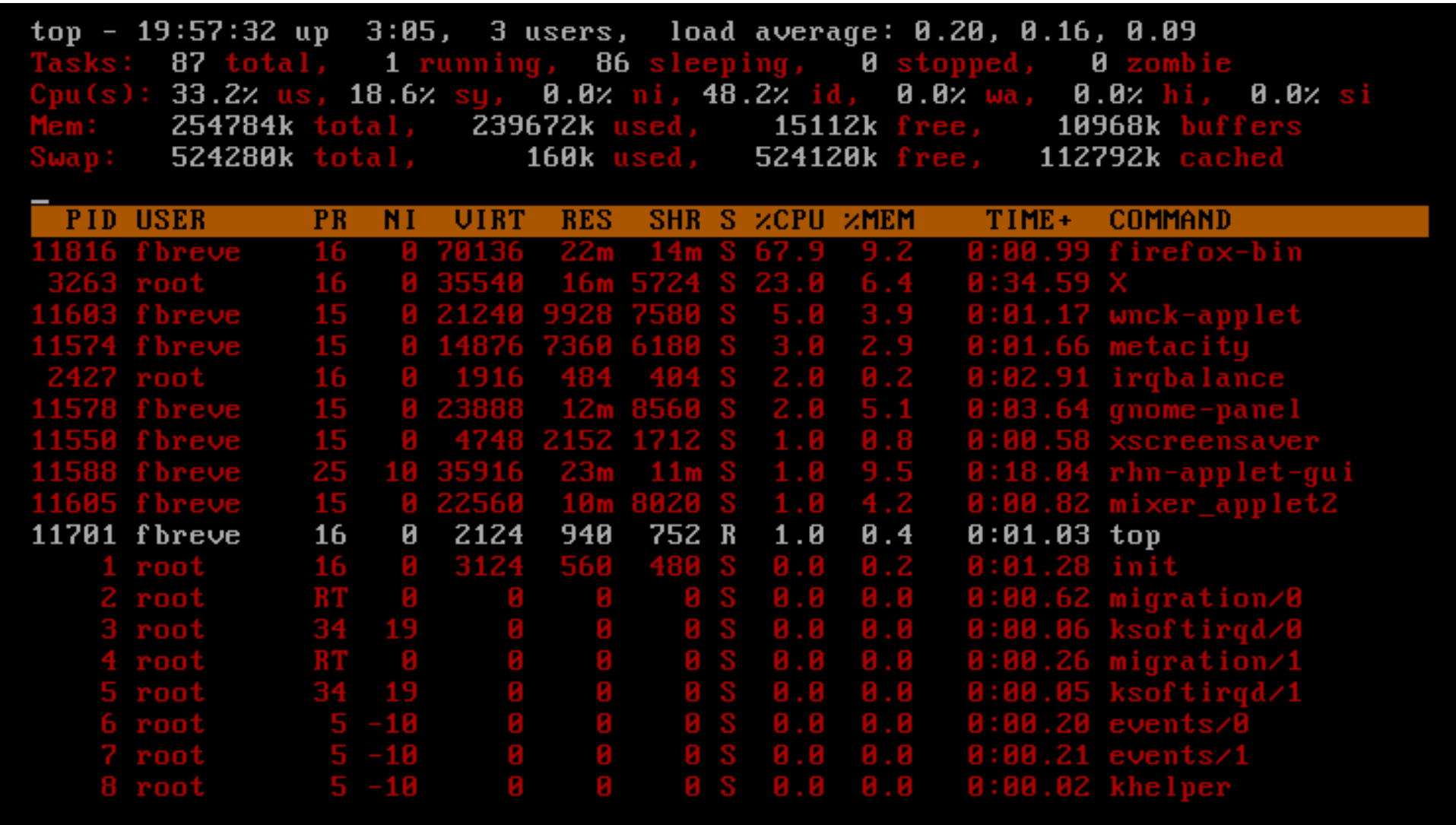
ps: Monitorando Processos

- ps recorta os comandos para caberem em uma linha
  - Para evitar esse truncamento use o argumento **w**
- Use o argumento **lax** para obter mais informações técnicas
  - Mais rápido, não traduz cada UID em nome
  - Inclui PPID (PID do pai), valor nice (NI) e nome ou função do kernel na qual o processo está dormente (WCHAN)

top: Monitoramento ainda melhor de processos

- O comando ps dá apenas um instântaneo de seu sistema
- O comando top fornece um sumário atualizado regularmente
  - Como padrão a tela é atualizada a cada 10 segundos
  - Processos mais ativos aparecem no alto
  - top consome recursos, portanto deve ser usado apenas para fins de diagnóstico

top


top - 19:57:32 up 3:05, 3 users, load average: 0.20, 0.16, 0.09
Tasks: 87 total, 1 running, 86 sleeping, 0 stopped, 0 zombie
Cpu(s): 33.2% us, 18.6% sy, 0.0% ni, 48.2% id, 0.0% wa, 0.0% hi, 0.0% si
Mem: 254784k total, 239672k used, 15112k free, 10968k buffers
Swap: 524280k total, 160k used, 524120k free, 112792k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
11816 fbreve 16 0 70136 22m 14m S 67.9 9.2 0:00.99 firefox-bin
3263 root 16 0 35540 16m 5724 S 23.0 6.4 0:34.59 X
11603 fbreve 15 0 21240 9928 7580 S 5.0 3.9 0:01.17 wnck-applet
11574 fbreve 15 0 14876 7360 6180 S 3.0 2.9 0:01.66 metacity
2427 root 16 0 1916 484 404 S 2.0 0.2 0:02.91 irqbalance
11578 fbreve 15 0 23888 12m 8560 S 2.0 5.1 0:03.64 gnome-panel
11550 fbreve 15 0 4748 2152 1712 S 1.0 0.8 0:00.58 xscreensaver
11588 fbreve 25 10 35916 23m 11m S 1.0 9.5 0:18.04 rhn-applet-gui
11605 fbreve 15 0 22560 10m 8020 S 1.0 4.2 0:00.82 mixer_applet2
11701 fbreve 16 0 2124 940 752 R 1.0 0.4 0:01.03 top
1 root 16 0 3124 560 480 S 0.0 0.2 0:01.28 init
2 root RT 0 0 0 0 S 0.0 0.0 0:00.62 migration/0
3 root 34 19 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/0
4 root RT 0 0 0 0 S 0.0 0.0 0:00.26 migration/1
5 root 34 19 0 0 0 S 0.0 0.0 0:00.05 ksoftirqd/1
6 root 5 -10 0 0 0 S 0.0 0.0 0:00.20 events/0
7 root 5 -10 0 0 0 S 0.0 0.0 0:00.21 events/1
8 root 5 -10 0 0 0 S 0.0 0.0 0:00.02 khelper

Sistema Linux de uma Universidade

```
top - 01:10:38 up 248 days,  9:10,  2 users,  load average: 0.13, 0.07, 0.02
Tasks:  44 total,   1 running,  43 sleeping,   0 stopped,   0 zombie
Cpu(s):   5.2% user,   8.6% system,   0.0% nice,  86.2% idle
Mem:     61744k total,    60284k used,     1460k free,     5984k buffers
Swap:   160640k total,     1240k used,   159400k free,    34828k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
21011 fabricio  14   0  1128 1128  896 R 13.8  1.8   0:17.46 top
    1 root       8   0   216  216  196 S  0.0  0.3   2:23.89 init
    2 root       9   0     0    0    0 S  0.0  0.0   0:00.11 keventd
    3 root      19  19     0    0    0 S  0.0  0.0   0:03.89 ksoftirqd_CPU0
    4 root       9   0     0    0    0 S  0.0  0.0   7:14.57 kswapd
    5 root       9   0     0    0    0 S  0.0  0.0   0:04.03 bdflush
    6 root       9   0     0    0    0 S  0.0  0.0   0:03.31 kupdated
   10 root      -1 -20     0    0    0 S  0.0  0.0   0:00.00 mdrecoveryd
   11 root       9   0     0    0    0 S  0.0  0.0   3:16.71 kjournald
   40 root       9   0     0    0    0 S  0.0  0.0   0:12.62 kjournald
   41 root       9   0     0    0    0 S  0.0  0.0   0:01.11 kjournald
   63 root       9   0   244  224  220 S  0.0  0.4  19:35.51 syslogd
   66 root       9   0    52    4    4 S  0.0  0.0   0:00.05 klogd
  409 root       9   0     0    0    0 S  0.0  0.0   0:00.00 khubd
  431 bin        9   0   104    4    4 S  0.0  0.0   0:00.89 rpc.portmap
  435 root       9   0    96    4    4 S  0.0  0.0   0:00.09 rpc.statd
  437 root       9   0     0    0    0 S  0.0  0.0   6:44.83 rpciod
```

Sistema Sun OS de outra Universidade

```
last pid:  4234;  load averages:  0.11,  0.07,  0.04                  00:38:20
69 processes:  66 sleeping, 1 zombie, 1 stopped, 1 on cpu
CPU states: 97.5% idle,  1.6% user,  0.9% kernel,  0.0% iowait,  0.0% swap
Memory: 2048M real, 1840M free, 33M swap in use, 2015M swap free

  PID USERNAME THR PRI NICE  SIZE   RES STATE   TIME    CPU COMMAND
 4234 fabricio   1  50    0 2608K 2184K cpu11   0:12  1.72% top
  549 apfreire   1  58    0 2320K 1120K sleep   0:15  0.01% lmgrd
  214      0     5  58    0 5248K 3168K sleep  60:07  0.00% automountd
  513      0    12  58    0 2744K 2032K sleep  19:47  0.00% mibiisa
  334      0     1  58    0 3600K  824K sleep  18:23  0.00% sshd
  246      0    11  58    0 9896K 8648K sleep   5:40  0.00% nscd
  170      0     3  58    0 2528K 1640K sleep   1:22  0.00% nis_cachemgr
  228      0    16  58    0 3800K 1952K sleep   0:39  0.00% syslogd
  510      0     1  58    0 2248K 1632K sleep   0:35  0.00% snmpdx
    1      0     1  58    0  856K  168K sleep   0:32  0.00% init
  168      0     7  30    0 3648K 1288K sleep   0:10  0.00% keyserv
  165      0     1  58    0 2464K 1560K sleep   0:06  0.00% rpcbind
26908 pvgf     1  58    0 6672K 2632K sleep   0:05  0.00% sshd
  194      0     1  58    0 2448K  720K sleep   0:05  0.00% inetd
  227      0     1  48    0 2512K 1760K sleep   0:03  0.00% cron
```

Processos descontrolados

- Tem duas variantes:
  - Processos de usuários que usam quantidade excessiva de um recurso
    - Tempo de CPU, espaço em disco, etc.
    - Não é necessariamente um defeito
  - Processos de sistema que de repente se enfurecem e apresentam um comportamento selvagem
    - Supostamente sempre devem se comportar de maneira razoável

Processos descontrolados

- Podemos identificar programas que usam muito tempo de CPU usando top
- Se estiver evidente que um processo está usando muito tempo devemos:
  - Entrar em contato com o dono do processo e perguntar a ele o que está acontecendo
  - Se não puder ser localizado teremos de investigar por contra própria
    - Embora normalmente não se deve investigar os diretórios dos usuários essa prática é aceitável quando queremos descobrir a fonte de um processo descontrolado.

Processos descontrolados

- Devemos sempre tentar descobrir o que está acontecendo
  - O processo pode ser legítimo e importante para o usuário
    - Não se deve eliminar processos apenas porque eles usam muitos recursos de da CPU
  - O processo pode ser mal intencionado ou destrutivo
    - Você precisa descobrir o que ele está fazendo
      - Quebrando senhas?

Processos descontrolados

- Se não for possível determinar a razão da existência do processo
  - Suspenda-o com STOP e envie e-mail para o proprietário explicando o que houve
    - Você poderá reiniciá-lo depois com CONT
    - Alguns processos poderão não ser reiniciados corretamente (conexões de rede não mais existentes, etc.)
- Se o processo parece estar fazendo algo razoável
  - Use renice para diminuir sua prioridade e solicite ao usuário que faça esse procedimento futuramente

Questões

- Suponha que um usuário em seu sistema tenha iniciado um processo de longa duração que está consumindo uma fração significativa dos recursos da máquina.
  - Como você reconheceria um processo que está exaurindo recursos?
  - Suponha que o processo possa ser legítimo e não deva ser extinto. O que você poderia fazer para colocá-lo na geladeira?
  - Mais tarde você descobre que o processo pertence a seu chefe e tem de continuar a ser processado. Como tirá-lo da geladeira?
  - Suponha que o processo tenha que ser extinto, que sinal você deveria emitir? Por que? E para garantir que ele fosse efetivamente extinto?

Questões

- Qual a diferença entre `top` e `ps`? Para que você deve executar cada um? Execute ambos e veja o que você pode determinar com um deles.

Referências Bibliográficas

- NEMETH, Evi.; HEIN, Trent R.; SNYDER, Garth. *Manual Completo do Linux: Guia do Administrador*. Makron Books, 2004.