Gestão de processos

José Pedro Oliveira
(jpo@di.uminho.pt)

Grupo de Sistemas Distribuídos
Departamento de Informática
Escola de Engenharia
Universidade do Minho

Sistemas Operativos 2005-2006

Conteúdo

Processos

Identificador de processo

pid - process identifier

ppid - parent process identifier

Estado de processos

D - uninterruptible sleep (IO),

R - runnable,

S - sleeping,

T - traced or stopped,

Z - zombie

Comando **ps**

Comando **ps**

O comando **ps** permite obter um snapshot dos processos actuais.

Synopsis

```
ps  [opções]
```

Algumas opções

- a - todos os processos sem terminal associado
- l - lista pids e ppids
- x - inclusive os processos sem terminal associado
- u - formato orientado ao utilizador
- -u user - processos do utilizador *user*
- f - hierarquia de processos am arte ASCII

Comando **ps**: exemplos

$ ps

```
  PID TTY          TIME CMD
 5549 pts/5    00:00:00 bash
 7530 pts/5    00:00:00 ps
```

Colunas

- PID - identificador do processo
- TTY - terminal associado ao processo
- TIME - tempo de CPU acumulado
- CMD - comando executado

Comando **ps**: exemplos

$ ps aux

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2064   552 ?        S    06:12   0:00 init [3]
root         2  0.0  0.0      0     0 ?        SN   06:12   0:00 [ksoftirqd/0]
root         3  0.0  0.0      0     0 ?        S<   06:12   0:00 [events/0]
...
```

Colunas

- USER - utilizador
- PID - identificador do processo
- TTY - terminal associado ao processo
- STAT - estado do processo
- TIME - tempo de CPU acumulado
- COMMAND - comando executado

Comando **ps**: exemplos

$ ps axf

```
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:00 init [3]
    2 ?        SN     0:00 [ksoftirqd/0]
    3 ?        S<     0:00 [events/0]
    4 ?        S<     0:00  \_ [khelper]
  ...
 4642 tty3     Ss+    0:00 /sbin/mingetty tty3
 4643 ?        Ss     0:00 login -- jpo
 5252 tty4     Ss     0:00  \_ -bash
 5299 tty4     S+     0:00      \_ /bin/sh /usr/X11R6/bin/startx
 5310 tty4     S+     0:00          \_ xinit /etc/X11/xinit/xinitrc --
 5311 ?        S      1:10              \_ X :0
 5384 tty4     S      0:00              \_ /bin/sh /usr/bin/startkde
 5774 tty4     S      0:00                  \_ kwrapper ksmserver
 4646 tty5     Ss+    0:00 /sbin/mingetty tty5
  ...
```

Comando **top**

Comando **top**

Permite monitorizar a actividade do sistema. Por omissão lista os processos com maior utilização de CPU.

Synopsis

```
top  [opções]
```

Algumas opções

-d - intervalo entre actualizações

-n - número de iterações

Comando **top**: exemplo

$ top

```
top - 02:25:32 up 53 min, 11 users,  load average: 0.04, 0.08, 0.08
Tasks:  91 total,   1 running,  90 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.0% us, 0.0% sy, 0.0% ni, 99.0% id, 0.0% wa, 0.0% hi, 0.0% si
Mem:    636984k total,   333860k used,   303124k free,    22140k buffers
Swap:  1020088k total,        0k used,  1020088k free,   185932k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5311 root      15   0  143m  10m 2196 S  0.3  1.7   0:43.71 X
 5809 jpo       15   0 32032  15m  12m S  0.3  2.6   0:02.84 kdeinit
 7652 jpo       17   0  2364  940  748 R  0.3  0.1   0:00.05 top
    1 root      16   0  2928  552  472 S  0.0  0.1   0:00.84 init
    2 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    3 root       5 -10     0    0    0 S  0.0  0.0   0:00.16 events/0
  ...
```

Comando **nice**

Comando **nice**

Permite executar programas com prioridades de escalonamento modificadas.

Synopsis

```
nice [opção] [comando [arg] ...]
```

Algumas opções

-n adjust - incrementa a prioridade em *adjust*

Comando **kill**

Comando **kill**

O comando kill permite enviar um sinal para um processo ou um grupo de processos. Se nenhum sinal for especificado, o sinal **TERM** é enviado por omissão. O sinal **TERM** mata todos os processos que não o interceptarem. Nalguns casos pode ser necessário enviar o sinal **KILL** (9), dado que este sinal não pode ser interceptado.

Synopsis

```
kill [opções] [pid] ...
```

Comando **kill**: listar sinais

```
$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1
36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4  39) SIGRTMIN+5
...
60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2  63) SIGRTMAX-1
64) SIGRTMAX
```

Comando **kill**: sinais

Alguns sinais

SIGHUP (1) - Hangup.

SIGINT (2) - Interrupt.
(gerado pela sequência de teclas CTRL+C)

SIGQUIT (3) - Quit.
(gerado pela sequência de teclas CTRL+\)

SIGKILL (9) - Kill, unblockable.
Este sinal não pode ser interceptado pela processo.

SIGTERM (15) - Termination.
Sinal enviado por omissão pelo comando **kill**.

Comando **kill**: exemplos

$ kill pid

Envia o sinal **TERM** para o processo *pid*. O processo ao receber este sinal deverá terminar a sua execução.

$ kill -9 pid

O sistema operativo termina o processo *pid*, não lhe dando qualquer hipótese de executar código de limpeza.

$ kill -INT pid

O sinal **INT** é enviado para o processo *pid*.

Conteúdo

Pseudo sistema de ficheiros /proc

Descrição

- O pseudo sistema de ficheiros /proc actua como interface a estruturas de dados do kernel.
- Pode ser utilizado para obter informação sobre o sistema e alterar certos parâmetros do kernel em *runtime* (sysctl).
- Enquanto que a grande maioria do pseudo sistema de ficheiros /proc só pode ser acedida para leitura, o ramo /proc/sys é utilizado para configurar o kernel em *runtime*, ou seja, permite operações de escrita.

Pseudo sistema de ficheiros /proc: informação

Informação

- subdirectórios por processo
- informação sobre o kernel
- dispositivos IDE (/proc/ide)
- informação sobre rede (/proc/net)
- informação SCSI (/proc/scsi)
- informação sobre porta paralela (/proc/parport)
- informação sobre TTY (/proc/tty)
- estatísticas diversas sobre o kernel (/proc/stat)

/proc/cpuinfo

Informação sobre o(s) processador(es)

$ cat /proc/cpuinfo

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 2
model name      : Intel(R) Pentium(R) 4 CPU 2.40GHz
stepping        : 4
cpu MHz         : 2405.473
cache size      : 512 KB
...
```

Processo nnn (/proc/nnn)

$ ls -l /proc/7218/

```
total 0
dr-xr-xr-x  2 jpo jpo 0 Feb 21 00:07 attr
-r--------  1 jpo jpo 0 Feb 21 00:09 auxv
-r--r--r--  1 jpo jpo 0 Feb 21 00:06 cmdline
lrwxrwxrwx  1 jpo jpo 0 Feb 21 00:09 cwd -> /home/users/jpo/csi
-r--------  1 jpo jpo 0 Feb 21 00:06 environ
lrwxrwxrwx  1 jpo jpo 0 Feb 21 00:09 exe -> /usr/X11R6/bin/gvim
dr-x------  2 jpo jpo 0 Feb 21 00:09 fd
-r--------  1 jpo jpo 0 Feb 21 00:09 maps
-rw-------  1 jpo jpo 0 Feb 21 00:09 mem
-r--r--r--  1 jpo jpo 0 Feb 21 00:09 mounts
lrwxrwxrwx  1 jpo jpo 0 Feb 21 00:09 root -> /
-r--r--r--  1 jpo jpo 0 Feb 21 00:06 stat
-r--r--r--  1 jpo jpo 0 Feb 21 00:09 statm
-r--r--r--  1 jpo jpo 0 Feb 21 00:06 status
dr-xr-xr-x  3 jpo jpo 0 Feb 21 00:09 task
-r--r--r--  1 jpo jpo 0 Feb 21 00:09 wchan
```

Processo nnn (/proc/nnn)

Informação sobre a linha de comando e as variáveis de ambiente

$ cat /proc/7218/cmdline | tr '\0' '\n'

```
gvim
processos.tex
```

$ cat /proc/7218/environ | tr '\0' '\n'

```
KDE_MULTIHEAD=false
HOSTNAME=localhost.localdomain
SHELL=/bin/bash
TERM=xterm
...
```

Processo nnn (/proc/nnn)

Informação sobre descriptores de ficheiros abertos

$ ls -l /proc/7218/fd

```
total 6
lrwx------  1 jpo jpo 64 Feb 21 00:18 0 -> /dev/pts/5
lrwx------  1 jpo jpo 64 Feb 21 00:18 1 -> /dev/pts/5
lrwx------  1 jpo jpo 64 Feb 21 00:18 2 -> /dev/pts/5
lrwx------  1 jpo jpo 64 Feb 21 00:18 3 -> socket:[17997]
lrwx------  1 jpo jpo 64 Feb 21 00:18 4 -> socket:[17999]
lrwx------  1 jpo jpo 64 Feb 21 00:18 6 -> /tmp/.processos.tex.swp
```

Configurar o kernel em *runtime*: /proc/sys

Introdução

O directório /proc/sys não só é uma fonte de informação, como permite alterar parâmetros do kernel.

Alterar parâmetros do kernel

Para alterar um valor basta apenas fazer `echo` do novo valor para o ficheiro.

Alterar um parâmetro do kernel

Exemplo

Alterar o comportamento da *stack* TCP/IP na recepção de pacotes ICMP broadcast.

Executar os seguintes comandos

- $ cd /proc/sys/net/ipv4
- $ cat icmp_echo_ignore_broadcasts
  0
- $ echo 1 > icmp_echo_ignore_broadcasts
- $ cat icmp_echo_ignore_broadcasts
  1

Alterar um parâmetro do kernel: comando sysctl

Comando **sysctl**

Permite configurar parâmetros em *runtime*. Os parâmteros disponiveis são os que se encontram listados em /proc/sys.

Synopsis

```
sysctl [opções] variável ...
sysctl [opções] -w variável=valor ...
...
```

Exemplos

- $ sysctl -n kernel.hostname
- $ sysctl -w kernel.hostname=posto123

Ficheiro de configuração /etc/sysctl.conf

$ cat /etc/sysctl.conf

```
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0
...
```

Referências

Referências

- man 8 sysctl
- man 5 sysctl.conf
- <kernel>/Documentation/filesystems/proc.txt

  Exemplo: linux-2.4.28/Documentation/filesystems/proc.txt