Introducción a Sistemas Operativos: Procesos

Clips xxx
Francisco J Ballesteros

1. /proc

Existe otro interfaz para manipular procesos más allá de las llamadas al sistema habituales para ello. Concretamente, hay un directorio en (la mayoría de los sistemas) UNIX que aparenta tener ficheros relacionados con procesos:

```
unix$ ls /proc
1      153    25   43    701        dma          mtrr
10     154    26   44    778        driver       net
1001   16     27   45    8          execdomains  pagetypeinfo
1002   17     28   46    89         fb           partitions
103    174    29   47    898        filesystems  sched_debug
1046   175    3    48    9          fs           schedstat
1071   18     30   5     90         interrupts   scsi
11     19     31   50    900        iomem        self
1130   2      32   516   903        ioports      slabinfo
12     20     325  52    908        irq          softirqs
13     21     33   522   909        kallsyms     stat
14     21102  330  53    911        kcore        swaps
143    21104  34   54    930        key-users    sys
144    21227  35   541   acpi       keys         sysrq-trigger
145    21246  36   55    asound     kmsg         sysvipc
146    21247  368  5627  buddyinfo  kpagecount   timer_list
147    21248  369  613   bus        kpageflags   timer_stats
148    21301  37   629   cgroups    loadavg      tty
149    21302  38   637   cmdline    locks        uptime
15     21317  39   67    consoles   mdstat       version
150    215    40   68    cpuinfo    meminfo      version_signature
151    22     41   69    crypto     misc         vmallocinfo
15160  23     42   7     devices    modules      vmstat
152    24     421  70    diskstats  mounts       zoneinfo
```

En este caso, hemos utilizado un sistema Linux. Los ficheros en `/proc` no son ficheros reales en el disco. UNIX se los inventa para reflejar el estado de los procesos que están ejecutando y para dejarte averigüar cosas sobre ellos e incluso operar sobre ellos. Aún más, no sólo para operar sobre procesos, sino para operar sobre el sistema entero.

Cuando un proceso lee o escribe uno de estos ficheros, UNIX hace que la operación sobre el fichero se comporte normalmente, pero UNIX inventa el resultado de la operación para hacer creer que dichos ficheros corresponden en cada momento al estado del sistema. Son ficheros *sintéticos*. Dicho de otro modo, son falsos y UNIX se los inventa en cada momento.

Como puedes ver por el listado de ficheros en `/proc` hay un directorio por proceso, siendo el nombre del directorio el pid del proceso. Dicho directorio contiene ficheros que permiten ver y cambiar datos del

proceso en cuestión. Por ejemplo, en un sistema Linux:

```
unix$ ps
  PID TTY          TIME CMD
21422 pts/0    00:00:00 bash
21438 pts/0    00:00:00 ps
unix$ ls -l /proc/21422
unix$ ls  /proc/21422
attr        coredump_filter  gid_map    mountinfo   oom_score      schedstat  status
autogroup   cpuset           io         mounts      oom_score_adj  sessionid  syscall
auxv        cwd              limits     mountstats  pagemap        setgroups  task
cgroup      environ          loginuid   net         personality    smaps      timers
clear_refs  exe              map_files  ns          projid_map     stack      uid_map
cmdline     fd               maps       numa_maps   root           stat       wchan
comm        fdinfo           mem        oom_adj     sched          statm
unix$
```

Hemos listado el directorio que corresponde al proceso del shell que estábamos ejecutando. Podemos ver cual es la línea de comandos para dicho proceso:

```
unix$ cat /proc/21422/cmdline
-bashunix$
```

Dado que era un shell de login (ejecutado al hacer el login en el sistema), el programa *login* que lo creó utilizó `-bash` como valor para el primer argumento (`argv[0]` en `main` en dicho proceso). El convenio en UNIX es que si en un shell, `argv[0]` comienza por "`-`", entonces se trata de un shell de login (y habitualmente dicho shell leerá `$HOME/.profile` u otro fichero para ejecutar los comandos que contenga como parte del proceso de inicialización).

En `/proc/21422/maps` tenemos una descripción de los segmentos de memoria que utiliza dicho proceso:

```
unix$ cat /proc/21422/maps
00400000-004ef000 r-xp 00000000 08:01 17039362                           /bin/bash
006ef000-006f0000 r--p 000ef000 08:01 17039362                           /bin/bash
006f0000-006f9000 rw-p 000f0000 08:01 17039362                           /bin/bash
006f9000-006ff000 rw-p 00000000 00:00 0
01f99000-021a0000 rw-p 00000000 00:00 0                                  [heap]
...
7f3164b43000-7f3164b44000 rw-p 00000000 00:00 0
7fff170f1000-7fff17112000 rw-p 00000000 00:00 0                          [stack]
unix$
```

Puedes ver las direcciones de comienzo y fin para los segmentos de texto, datos inicializados de sólo lectura, datos inicializados, BSS, y pila (entre otros). Hemos borrado del listado los segmentos correspondientes a las librerías dinámicas que utiliza dicho proceso. Resulta interesante ver que los segmentos de texto y datos inicializados proceden de `/bin/bash` para este proceso. Como verás, el texto tiene permiso de ejecución y los datos tienen permiso de lectura al menos. Los datos que no son constantes están en un segmento con permisos de lectura escritura.

A la vista de esto, aunque en C, `"hola"` es un array de `char`, no deberías intentar cambiar su contenido. Es muy posible que dicho array esté guardado en memoria en un segmento de datos de sólo lectura durante la ejecución del programa.

El fichero `/proc/21422/status` tiene información interesante sobre el estado del proceso:

```
unix$ cat /proc/21422/status
Name:    bash
State:    S (sleeping)
Pid:    21422
PPid:    21421
...
voluntary_ctxt_switches:    398
nonvoluntary_ctxt_switches:    247
unix$
```

Entre otras cosas, puedes ver que actualmente el proceso está bloqueado (durmiendo) y que unas 398 veces ha hecho llamadas al sistema que han provocado que se le expulse del procesador. Además, unas 247 veces ha sido UNIX el que lo ha expulsado sin que el proceso hiciera ninguna llamada que causara la expulsión. Simplemente, llegaría una interrupción de reloj y UNIX decidiría que ya era hora de empezar a ejecutar un rato otro proceso.

Recuerda que `/proc` (en la mayoría de los UNIX) es una abstracción y no corresponde a datos reales guardados en ningún disco. UNIX se inventa esos ficheros respondiendo a las llamadas al sistema que operan sobre dichos ficheros para que aparentemente `/proc` contenga una representación del estado de los procesos y el sistema.