

## Linux e Processos

---

Um processo é um programa em execução. Em um sistema operacional, entender e interpretar as ações dos processos é de fundamental importância para saber o que está sendo executado. Cada processo iniciado pelo sistema operacional tem um identificador numérico PID (Process ID). A maneira mais fácil de criar um processo no Linux é carregar o programa através da interface da linha de comando ou através da interface gráfica. Por exemplo, na linha de comando ou Shell:

```
$ ls
```

O Shell recebe o comando como argumento e cria um novo processo que executa o programa `ls`, que lista o conteúdo do diretório corrente.

Linux é um sistema multitarefa que permite a execução concorrente de vários processos. Para listar os processos em execução no sistema, execute o comando `ps` na linha de comando:

```
$ ps
  PID TTY          TIME CMD
 25547 pts/0    00:00:00 bash
 27807 pts/0    00:00:00 ps
$
```

Os campos mostrados são:

**PID** - o identificador do processo

**TTY** - o terminal que o processo está ligado (mais sobre isso depois)

**TIME** - a quantidade de tempo gasto pelo processo no processador.

**CMD** - o nome do programa executado.

Vamos entender o que está acontecendo: há um processo (o shell) com o qual você dialoga a fim de executar comandos. Esse é o processo de número 25547. O comando solicitado consiste na execução do programa ps. Esse programa, ao ser executado, descobre que neste momento há dois processos: ele próprio (de número 27807) e o shell.

Na verdade, além desses dois pode haver muitos outros que o ps não exhibe. Através das opções aux entretanto pode-se exibir todos os processos correntes e com vários campos de descrição.  
(não mostramos todos aqui. Execute o “ps aux” na sua máquina Linux)

```
$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.3	103260	13020	?	Ss	17:32	0:02	/sbin/init
root	2	0.0	0.0	0	0	?	S	17:32	0:00	[kthreadd]
...										
lauro	2947	0.0	0.1	10620	4996	pts/0	Ss	18:32	0:00	bash
lauro	3296	0.0	0.0	11480	3256	pts/0	R+	18:54	0:00	ps aux

Neste caso, todos os processos além do shell e o ps dizem respeito apenas à administração do sistema. O /sbin/init (por exemplo) é o processo inicial que o SO gera e que cria todos os outros processos de usuário. O processo [kthreadd] é a thread inicial do kernel, que cria todas as outras threads do kernel. Normalmente, os nomes entre [ ] são threads do kernel (obs.: veremos mais a frente a relação entre processos e threads no Linux)

O número de um processo (PID) é usado para identificá-lo dentre os outros, por exemplo quando é necessário interromper prematuramente a sua execução. O unix vai numerando os processos em ordem crescente, à medida em que vão sendo criados.

## OPÇÕES DO PS

O comando ps possui muitas opções. Vamos mostrar apenas as opções no estilo BSD, outras podem ser encontradas através das páginas de manual (man ps):

**ps x** - mostra todos os processos que o seu usuário é dono.

**ps aux** - mostra todos os processos executando no sistema.

Outros campos incluídos na listagem de "ps aux" são:

**USER** - O nome do dono do processo.

**%CPU** - O tempo de execução no processador dividido pelo tempo total no sistema em porcentagem. (cputime/realtime ratio).

**%MEM** - RSS/(memória física total) em porcentagem

**VSZ** - Tamanho da memória virtual do processo em unidades de 1024 bytes (code+data+stack)

**RSS** - (Resident Set Size) tamanho de memória física usada pelo processo.

**START** - quando o processo iniciou.

**STAT** - o estado e outras características do processo no neste momento.

Códigos principais:

**I** - thread do kernel interrompida.

**R** - executando.

**S** - esperando um evento ocorrer.

**Z** - Processo no estado "Zombie" (terminou mas o pai não estava esperando).

**T** - Parado pelo controle de processo do shell.

## **MATANDO PROCESSOS**

Para matar um processo, envie um "sinal" com o comando kill. Um "sinal" é uma mensagem que pode ser enviada para um processo através do kernel.

```
$ kill <signal> <pid>
```

Existem vários tipos de sinais. O padrão é o TERM, ou o sinal de término.

Envie outros sinais adicionando uma opção ao kill. Por exemplo, para pausar um processo, use o sinal STOP:

```
$ kill -STOP 3296
```

O processo "pausado" anda está na memória, pronto para executar de onde parou. Use o sinal CONT para voltar a executar o processo:

```
$ kill -CONT 3296
```

O sinal mais radical para terminal um processo é o KILL. Outros sinais permitem que o processo trate o sinal, mas o KILL simplesmente remove o processo da memória. Use este sinal como último recurso.

Para conhecer outros sinais, use a opção -l:

```
$ kill -l
```

## CONTROLE DE PROCESSOS

O shell possui comandos de controle dos processos.

Você pode parar um processo com o CTRL-Z, e daí iniciá-lo novamente com o comando fg ou o comando bg (inicie no "background").

Para listar os processos executando no "background", execute o comando jobs.

```
$ du -h / > du.txt 2>&1
^Z
[1]+  Stopped                  du -h / > du.txt 2>&1
$ bg
[1]+  du -h / > du.txt 2>&1 &
$ jobs
[1]+  Running                  du -h / > du.txt 2>&1 &
$ fg
du -h / > du.txt 2>&1
$
```

Você pode iniciar o comando no background usando o caracter &.

```
$ du -h / > du.txt 2>&1 &
[1] 2909
$
```

Shell mostra a linha [1] 2909 indicando o job número 1 e o pid do processo criado.

O comando do exemplo está redirecionando a saída padrão e a saída de erro padrão do terminal (2>&1) para o arquivo du.txt. Veremos mais sobre isso em sistemas de arquivos.

## Monitorando os processos

O comando **ps** lista os processos correntes, mas não acompanha como o processo se comporta no tempo. Assim, é difícil determinar qual processo está usando muito o processador ou memória.

O comando **top** é mais útil porque mostra o status corrente do sistema e atualiza os valores a cada segundo. Também localiza no topo da lista aqueles processos mais ativos:

\$ top

## Unix top command

```
top - 16:17:31 up 127 days, 4:38, 5 users, load average: 0.21, 0.19, 0.13
Tasks: 123 total, 1 running, 81 sleeping, 0 stopped, 0 zombie
%Cpu(s): 4.7 us, 2.2 sy, 0.0 ni, 93.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4039588 total, 137876 free, 2264716 used, 1636996 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 2006124 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
13468	mongodb	20	0	1414808	362516	416	S	3.7	9.0	1808:54	mongod
18315	root	20	0	3299048	622164	26004	S	1.0	15.4	3419:47	influxd
7081	schkn	20	0	715428	432856	16260	S	0.7	10.7	53:38.80	prometheus
22433	schkn	20	0	107984	5688	4684	S	0.7	0.1	0:01.25	sshd
23616	telegraf	20	0	1200720	20436	6660	S	0.7	0.5	389:22.88	telegraf
17644	schkn	20	0	113132	14432	7880	S	0.3	0.4	0:06.95	pushgateway
27819	schkn	20	0	44544	3908	3292	R	0.3	0.1	0:00.61	top
1	root	20	0	225544	6812	4188	S	0.0	0.2	3:33.77	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.28	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	2:20.14	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	71:32.57	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.74	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:23.83	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.01	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.01	cpuhp/1
14	root	rt	0	0	0	0	S	0.0	0.0	0:25.11	watchdog/1
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.73	migration/1
16	root	20	0	0	0	0	S	0.0	0.0	1:47.74	ksoftirqd/1
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs

Como se pode ver pela imagem anterior, o comando “top” devolve informações sobre:

- Utilização do CPU;
- Informação sobre o “top” dos processos;
- Utilização da memória RAM;
- Número de utilizadores logados no sistema;
- Informação sobre a memória SWAP;
- Número de tarefas etc.

Você pode usar o top com vários comandos (no teclado):

espaço	Atualiza os valores imediatamente
M	Ordena pelos valores de uso da memória residente.
T	Ordena pelo uso do total (cumulativo) do processador.
P	Ordena pelo uso corrente do processador (padrão)
U	Mostra apenas os processos do usuário.
F	Seleciona outros campos para mostrar.
?	Mostra um sumário dos comandos.

O comando htop é uma evolução do comando top. As informações são semelhantes às produzidas pelo comando top, mas apresentadas num interface mais intuitiva e colorida. Para executar o htop basta abrir a linha de comandos e escrever:

```
$ htop
```

## Gauges

## Unix htop command

```

1  [[ 2.7%] Tasks: 59, 203 thr; 1 running
2  [[ 3.3%] Load average: 0.12 0.14 0.10
Mem[|||||||||||||||||||||||||||||||||2.19G/3.85G] Uptime: 127 days(!), 04:48:17
Swp[|||||0K/0K]

PID USER      PRI  NI  VIRT   RES   SHR   S  CPU% MEM%   TIME+  Command
7081 schkn      20    0  698M  422M 16260 S   0.7 10.7 53:43.93 ./prometheus
13561 mongod     20    0 1381M 353M  416 S   0.7  9.0 3h56:14 /usr/bin/mongod --unixSocketPrefix=/run/mongod --config
16589 schkn     20    0  969M 65284  846 S   0.7  1.6 2h27:30 node /home/schkn/code/chrome-keyboard/website/index.js
23904 telegraf  20    0 1172M 20276 6660 S   0.7  0.5 28:39.43 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -c8
324 root       20    0 3221M  639M 25992 S   0.7 16.2 7h31:18 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
1378 schkn     20    0 32408  4408 3572 R   0.0  0.1 0:02.52 http
17644 schkn     20    0  110M 14432  7880 S   0.0  0.4 0:10.28 ./pushgateway
7087 schkn     20    0  698M  422M 16260 S   0.0 10.7 7:21.27 ./prometheus
17514 schkn     20    0 105M   5412  4408 S   0.0  0.1 0:03.10 sshd: schkn@pts/0
17664 schkn     20    0  110M 14432  7880 S   0.0  0.4 0:02.40 ./pushgateway
1628 schkn     20    0  698M  422M 16260 S   0.0 10.7 6:25.84 ./prometheus
18315 root       20    0 3221M  639M 25992 S   0.0 16.2 5h59:51 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
23616 telegraf  20    0 1172M 20276 6660 S   0.0  0.5 6h29:24 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -co
22433 schkn     20    0  105M 5688  4684 S   0.0  0.1 0:02.10 sshd: schkn@pts/3
13468 mongod     20    0 1381M 353M  416 S   0.0  9.0 3h08:58 /usr/bin/mongod --unixSocketPrefix=/run/mongod --config
17396 kapacitor 20    0 2060M  231M 11204 S   0.0  5.9 32h18:41 /usr/bin/kapacitor -config /etc/kapacitor/kapacitor.conf
7084 schkn     20    0  698M  422M 16260 S   0.0 10.7 8:37.44 ./prometheus
17649 schkn     20    0  110M 14432  7880 S   0.0  0.4 0:02.27 ./pushgateway
18376 root       20    0 3221M  639M 25992 S   0.0 16.2 4h00:09 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
27414 root       20    0 3221M  639M 25992 S   0.0 16.2 3h44:11 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
28225 kapacitor 20    0 2060M  231M 11204 S   0.0  5.9 2h40:38 /usr/bin/kapacitor -config /etc/kapacitor/kapacitor.conf
23670 telegraf  F3Search 1172M 20276 6660 S   0.0  0.5 40:08.94 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -co

```

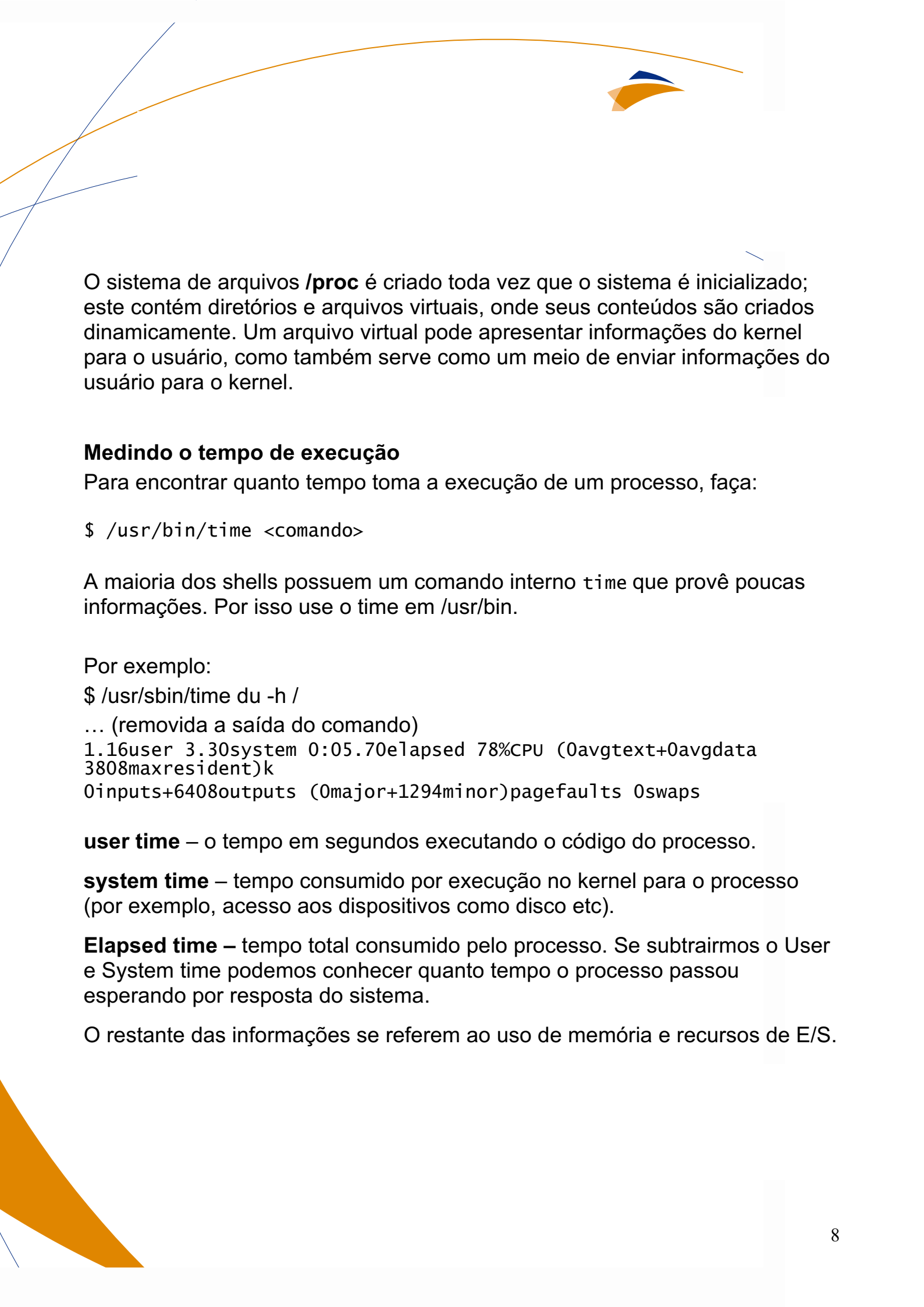
- Individual process details

## O Sistema de arquivos /proc

A maioria dos sistemas operacionais baseados em Unix possuem um sistema de arquivos **/proc**, que fornece uma maneira fácil de acesso às informações sobre os processos, atuando como uma interface para estrutura de dados internas do kernel e possibilitando a alteração de parâmetros do kernel em tempo de execução.

O sistema de arquivos **/proc** oferece muitas possibilidades de interação com as informações internas do sistema, como por exemplo:

- visualização de informações de estatísticas;
- visualização de informações de hardware;
- alteração de parâmetros em tempo de execução;
- visualização e modificação dos parâmetros de rede e host;
- visualização de informações de desempenho da memória.



O sistema de arquivos **/proc** é criado toda vez que o sistema é inicializado; este contém diretórios e arquivos virtuais, onde seus conteúdos são criados dinamicamente. Um arquivo virtual pode apresentar informações do kernel para o usuário, como também serve como um meio de enviar informações do usuário para o kernel.

### Medindo o tempo de execução

Para encontrar quanto tempo toma a execução de um processo, faça:

```
$ /usr/bin/time <comando>
```

A maioria dos shells possuem um comando interno `time` que provê poucas informações. Por isso use o `time` em `/usr/bin`.

Por exemplo:

```
$ /usr/sbin/time du -h /
```

... (removida a saída do comando)

```
1.16user 3.30system 0:05.70elapsed 78%CPU (0avgtext+0avgdata  
3808maxresident)k  
0inputs+6408outputs (0major+1294minor)pagefaults 0swaps
```

**user time** – o tempo em segundos executando o código do processo.

**system time** – tempo consumido por execução no kernel para o processo (por exemplo, acesso aos dispositivos como disco etc).

**Elapsed time** – tempo total consumido pelo processo. Se subtrairmos o User e System time podemos conhecer quanto tempo o processo passou esperando por resposta do sistema.

O restante das informações se referem ao uso de memória e recursos de E/S.



