

Administração de Sistemas Operacionais Linux , Continuação de Controle de processos

Antonio Rafael Braga

Universidade Federal do Ceara

11 de julho de 2025

../images/slide_capa_169.jpg

Agenda

- 1 nice e renice: prioridade de escalonamento de influência
- 2 O sistema de arquivo /proc
- 3 strace: trace sinais e chamadas de sistema
../Aula03/images/bgufc169.png
- 4 Processos periódicos
- 5 Atividade Prática

nice e renice

- A “*niceness*” de um processo é um valor numérico para o kernel sobre como o processo deve ser tratado em relação a outros processos que disputam a CPU.
- Um valor alto de “*niceness*” significa uma baixa prioridade para seu processo.
- Um valor baixo ou negativo significa alta prioridade: você não está sendo “nice”.
- Não é usual setar prioridades hoje em dia devido ao alto poder computacional de hoje!

nice e renice

- A “*niceness*” de um processo pode ser definida no momento da criação com o comando **nice** e ajustada posteriormente com o comando **renice**.
- **nice** usa uma linha de comando como argumento, e **renice** usa um PID ou (às vezes) um nome de usuário.

../Aula03/images/bgurc169.png

Exemplos

\$ nice -n 5 /bin/longtask // prioridades mais baixas (elevam o nice) por 5

\$ sudo renice -5 8829 // Configura niceness para -5

\$ sudo renice 5 -u boggs // Configura niceness do processo do boggs para 5

O sistema de arquivo /proc

- **/proc** é um pseudo sistema de arquivos na qual o kernel coleta uma variedade de informações do estado do sistema;
- **ps** e **top** leem as informações de status dos seus processos do **/proc**
No momento de sua criação (via chamada de sistema)
../Aula03/images/bgufc169.png
- As informações específicas do processo são divididas em subdiretórios nomeados por PID. Por exemplo, **/proc/1** é sempre o diretório que contém informações sobre o **init**.

Arquivos de informação do processo no Linux /proc (sub-diretórios numerados)

File	Contents
cgroup	The control groups to which the process belongs
cmd	Command or program the process is executing
cmdline^a	Complete command line of the process (null-separated)
cwd	Symbolic link to the process's current directory
environ	The process's environment variables (null-separated)
exe	Symbolic link to the file being executed
fd	Subdirectory containing links for each open file descriptor
fdinfo	Subdirectory containing further info for each open file descriptor
maps	Memory mapping information (shared segments, libraries, etc.)
ns	Subdirectory with links to each namespace used by the process.
root	Symbolic link to the process's root directory (set with chroot)
stat	General process status information (best decoded with ps)
statm	Memory usage information

strace: trace sinais e chamadas de sistema

- Difícil verificar o que um processo está fazendo atualmente;
- Você pode espionar o processo em baixo nível com o **strace**;
- Ele exibe todas as chamadas de sistema que um processo faz e todos os sinais que recebe;

strace: exemplo

\$ sudo strace -p 868

```
jeandro@cerf:~$ sudo strace -p 868
strace: Process 868 attached
pselect6(9, [3 4 5 6 7 8], NULL, NULL, {tv_sec=1, tv_nsec=24912596}, {sigmask=[], sigsetsize=8}) = 0 (Timeout)
newfstatat(AT_FDCWD, "/home/jeandro/.gnupg", {st_mode=S_IFDIR|0700, st_size=4096, ...}, 0) = 0
pselect6(9, [3 4 5 6 7 8], NULL, NULL, {tv_sec=4, tv_nsec=616775}, {sigmask=[], sigsetsize=8}) = 0 (Timeout)
newfstatat(AT_FDCWD, "/home/jeandro/.gnupg", {st_mode=S_IFDIR|0700, st_size=4096, ...}, 0) = 0
pselect6(9, [3 4 5 6 7 8], NULL, NULL, {tv_sec=4, tv_nsec=459288}, {sigmask=[], sigsetsize=8}) = 0 (Timeout)
newfstatat(AT_FDCWD, "/home/jeandro/.gnupg", {st_mode=S_IFDIR|0700, st_size=4096, ...}, 0) = 0
pselect6(9, [3 4 5 6 7 8], NULL, NULL, {tv_sec=4, tv_nsec=520120}, {sigmask=[], sigsetsize=8}) = 0 (Timeout)
newfstatat(AT_FDCWD, "/home/jeandro/.gnupg", {st_mode=S_IFDIR|0700, st_size=4096, ...}, 0) = 0
pselect6(9, [3 4 5 6 7 8], NULL, NULL, {tv_sec=4, tv_nsec=503644}, {sigmask=[], sigsetsize=8}) = 0 (Timeout)
newfstatat(AT_FDCWD, "/home/jeandro/.gnupg", {st_mode=S_IFDIR|0700, st_size=4096, ...}, 0) = 0
pselect6(9, [3 4 5 6 7 8], NULL, NULL, {tv_sec=4, tv_nsec=420757}, {sigmask=[], sigsetsize=8}) = 0 (Timeout)
newfstatat(AT_FDCWD, "/home/jeandro/.gnupg", {st_mode=S_IFDIR|0700, st_size=4096, ...}, 0) = 0
pselect6(9, [3 4 5 6 7 8], NULL, NULL, {tv_sec=4, tv_nsec=516997}, {sigmask=[], sigsetsize=8}) = 0 (Timeout)
rt_sigprocmask(SIG_BLOCK, ~[], [HUP INT USR1 USR2 TERM], 8) = 0
clone(child_stack=0x7f4b4dc69ef0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARPID,
rt_sigprocmask(SIG_SETMASK, [HUP INT USR1 USR2 TERM], NULL, 8) = 0
newfstatat(AT_FDCWD, "/home/jeandro/.gnupg", {st_mode=S_IFDIR|0700, st_size=4096, ...}, 0) = 0
futex(0x7f4b4e149060, FUTEX_WAKE_PRIVATE, 1) = 1
pselect6(9, [3 4 5 6 7 8], NULL, NULL, {tv_sec=4, tv_nsec=1513425}, {sigmask=[], sigsetsize=8}) = 1 (in [4], left {tv_sec=4, tv_nsec=1502334})
```

Processos que consomem muitos recursos

- Processos de "fuga" são aqueles que absorvem significativamente mais recursos de CPU, disco ou rede do sistema do que sua função ou comportamento normal poderia levar você a esperar;
- comandos **fuser** e **lsof** ajudam a identificar arquivos e mais informações do que os processos estão utilizando.

../Aula03/images/bgufc169.png

cron: comandos para escalonamento

- O *daemon* **cron** é a ferramenta tradicional para executar comandos em uma programação predeterminada;
- O **cron** lê os arquivos de configuração que contêm listas de linhas de comando e horários em que devem ser invocados;
- Um arquivo de configuração cron é chamado de “crontab”;

../Aula03/images/bgufc169.png


O formato dos arquivos crontab

minute hour dom month weekday command

Especificações de tempo do crontab

Field	Description	Range
<i>minute</i>	Minute of the hour	0 to 59
<i>hour</i>	Hour of the day	0 to 23
<i>dom</i>	Day of the month	1 to 31
<i>month</i>	Month of the year	1 to 12
<i>weekday</i>	Day of the week	0 to 6 (0 = Sunday)

Atividade Prática

- 1 Escolha uma das informações geradas pelo comando `top`, e identifique a qual arquivo no `/proc` estão essas informações.
- 2 Identifique um subdiretório no `/proc` e descreva quais informações você consegue visualizar. Ilustre com um print de tela.

- 3 Utilize o comando **strace** em algum processo do seu cliente e descreva quais informações você conseguiu identificar.

Atividade Prática 2

- 1 Abra um editor de texto e em outra aba execute o *strace* para o processo gerado pelo editor. Digite algum texto e salve o arquivo. Identifique e defina as chamadas de sistema utilizadas.

../Aula03/images/bgufc169.png

Referências

../Aula03/images/bgufc169.png