

Caio Rangel Ferreira Rodrigues  
Guilherme Almeida Lopes  
Ingrid Reupke Sbeguen Moran

## **Laboratório 2: Manipulação de Processos**

Relatório técnico de laboratório solicitado  
pelo professor Rodrigo Campiolo na disci-  
plina de Sistemas Operacionais do Bachare-  
lado em Ciência da Computação da Univer-  
sidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR  
Departamento Acadêmico de Computação – DACOM  
Bacharelado em Ciência da Computação – BCC

Campo Mourão  
Agosto / 2023

# Resumo

Este documento descreve a identificação e manipulação dos processos em Linux. Os comandos apresentam explicações detalhadas e imagens para exemplificação.

**Palavras-chave:** sistemas operacionais. linux. processos. programas.

# Sumário

1	Introdução . . . . .	4
2	Fundamentação . . . . .	4
3	Materiais . . . . .	4
4	Procedimentos e Resultados . . . . .	4
4.1	Questão 1 . . . . .	4
4.2	Questão 2 . . . . .	6
4.3	Questão 3 . . . . .	6
4.4	Questão 4 . . . . .	8
4.5	Questão 5 . . . . .	8
5	Discussão dos Resultados . . . . .	8
6	Conclusões . . . . .	9
7	Referências . . . . .	9

# 1 Introdução

Neste documento consta a fundamentação do conteúdo apresentado, os materiais utilizados para a realização do laboratório, os procedimentos efetuados e seus resultados, a discussão de tais resultados obtidos e, por fim, a conclusão.

## 2 Fundamentação

Um processo, composto por código, dados e contexto de execução, é uma aplicação que está em execução (TANENBAUM; BOS, 2016). Processos podem ser "pais", ou seja, eles são capazes de criar processos filhos que são derivados de tal "pai". Além disso, é importante ter conhecimento de como gerenciar os processos, ou seja, criá-los, bloqueá-los, suspendê-los, finalizá-los, entre outros, utilizando os sinais disponíveis. É possível também alterar a prioridade de um processo com o comando nice. Ademais, é importante entender que os processos zombies surgem no sistema operacional a partir da morte de um processo pai sem haver a remoção dos processos filhos que também já estão mortos. Tais processos filhos, já finalizados, porém não eliminados, são denominados processos zumbis.

## 3 Materiais

- Linux Debian 11
- Linguagem de programação C
- Visual Studio Code

## 4 Procedimentos e Resultados

### 4.1 Questão 1

Ao executar o comando `ps aux`, é possível identificar três programas do sistema - ou daemons - (Figuras 1, 2 e 3) e três programas do usuário (Figuras 4, 5 e 6). Para entender melhor, é necessário ter conhecimento dos valores que cada coluna representa. A primeira coluna (USER) se trata do usuário que é dono de tal processo, por exemplo, o primeiro programa do sistema apresentado foi iniciado pelo usuário `avahi` e o primeiro programa do usuário apresentado foi iniciado pelo usuário `Guilherme`, ou seja, o usuário da máquina. Logo após, existe o PID, abreviação para Process ID, em outras palavras, o número que identifica cada processo e diferencia eles. Utilizando os mesmos 2 exemplos citados anteriormente, exibem-se os PID's 600 e 90653. Nas próximas colunas, %CPU, %MEM,

VSZ e RSS, é mostrado, respectivamente, o tempo de CPU dividido pelo tempo que o processo está ativo, informações sobre a memória física, a quantidade de memória virtual utilizada e a memória usada em RAM. O TTY indica o terminal que controla o processo. A coluna STAT, por sua vez, exibe o estado do processo, que pode ser R (Running), Z (Zombie), I (Idle Task), S (Interruptible Sleep), D (Uninterruptible Sleep) e T (Stopped). Nos exemplos dados anteriormente, ambos estão no estado S (Ss e Sl), ou seja, estão esperando por um evento. As colunas START e TIME, representam, respectivamente, o horário em que o processo foi iniciado e o tempo de CPU acumulado. Por último, é possível visualizar o COMMAND, que mostra o nome do comando do processo.

```
root      595  0.0  0.0 239232 2144 ?        Ssl  ago31  0:00 /usr/libexec/
avahi     600  0.0  0.0   9364 1060 ?        Ss   ago31  0:01 avahi-daemon:
root      604  0.0  0.0  22828 1536 ?        Ss   ago31  0:00 /usr/libexec/
```

Figura 1 – Programa do sistema (PID 600).

```
Guilher+ 2630  0.0  0.1 472436 4732 ?        Sl   set01  0:02 /usr/bin/python3 /usr/bin/blueman-applet
Guilher+ 2653  0.0  0.0 384804 3572 ?        Ssl  set01  0:01 /usr/bin/zeitgeist-daemon
Guilher+ 2662  0.0  0.1 273144 6492 ?        Ssl  set01  0:00 /usr/lib/zeitgeist/zeitgeist-fts
```

Figura 2 – Programa do sistema (PID 2653).

```
rtkit     1272  0.0  0.0 155712 232 ?        SNsl ago31  0:01 /usr/libexec/rtkit-daemon
root      1343  0.0  0.0      0  0 ?        I<   ago31  0:00 [iprt-VBoxWQueue]
root      1351  0.0  0.0      0  0 ?        S    ago31  0:00 [iprt-VBoxTscThr]
```

Figura 3 – Programa do sistema (PID 1272).

```
Guilher+ 90653 0.3 1.6 2388308 64012 ?      Sl   23:08  0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 206 -isForBrowser -prefsLen
Guilher+ 90796 0.0 0.1 11716 4160 pts/0    R+   23:08  0:00 ps aux
```

Figura 4 – Programa do usuário (PID 90653).

```
Guilher+ 89840 0.3 1.2 477812 48948 ?        Ssl  22:21  0:08 /usr/libexec/gnome-terminal-server
Guilher+ 89847 0.0 0.1 10236 5788 pts/0      Ss   22:21  0:00 bash
root      89855 0.0 0.0      0  0 ?        T    22:21  0:00 [kworker/1:1-events]
```

Figura 5 – Programa do usuário (PID 89840).

```
Guilher+ 2139 1.8 0.2 2994284 10372 ?      S<sl set01  70:23 /usr/bin/pulseaudio --daemonize=no --log-target=journal
Guilher+ 2141 0.0 0.2 585664 10076 ?        SNsl set01  0:06 /usr/libexec/tracker-miner-fs
Guilher+ 2145 0.0 0.0 11472 2868 ?          Ss   set01  0:09 /usr/bin/dbus-daemon --session --address=systemd: --nofork
```

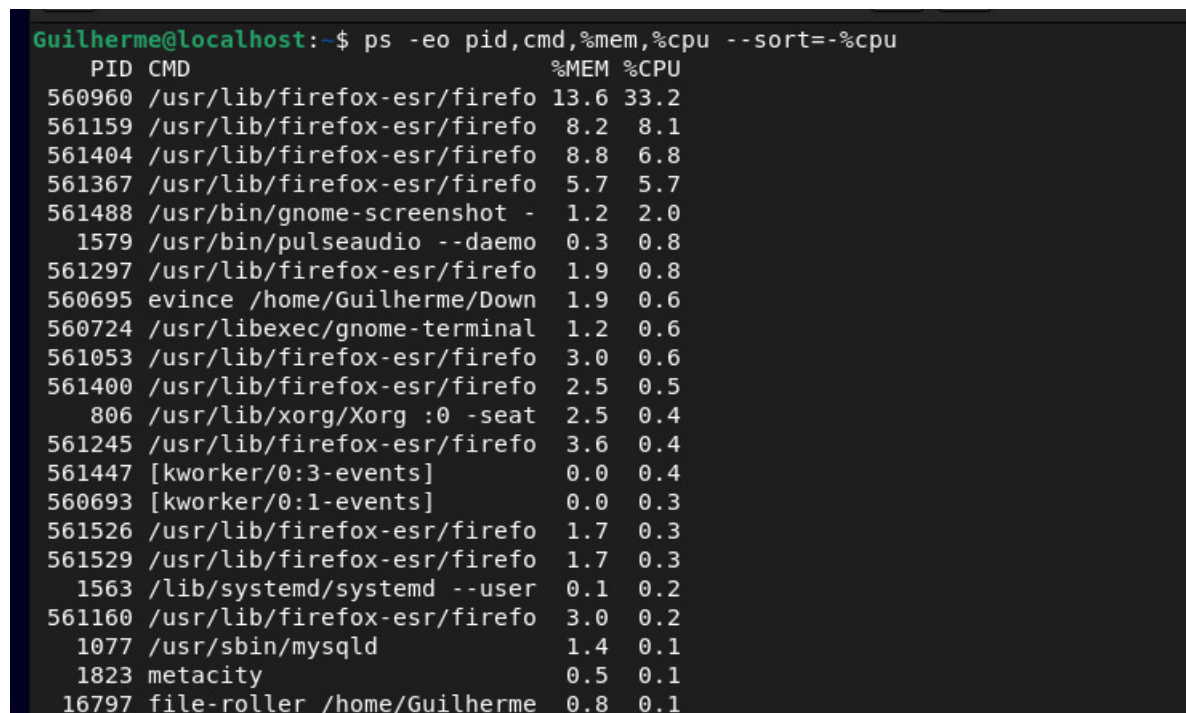
Figura 6 – Programa do usuário (PID 2141).

## 4.2 Questão 2

No sistema operacional utilizado, não havia nenhum processo zombie, que podem ser identificados pelo estado (STAT) Z. Não é possível eliminar tais processos do sistema a partir do comando `kill -SIGKILL pid_zombie`, pois tal sinal é utilizado para matar os processos, e os processos zumbis já estão mortos, uma vez que eles são apenas a estrutura do processo no qual não há um código executando. Dito isso, só seria possível eliminá-los desalocando a estrutura a partir da eliminação do processo pai.

## 4.3 Questão 3

Ao executar os comandos `ps -eo pid,cmd,%mem,%cpu --sort=-%cpu` (Figura 7), `ps -eo pid,cmd,%mem,%cpu --sort=-%mem` (Figura 8) e `ps -eo pid,user,cmd,time,%mem,%cpu --sort=-%time` (Figura 9) é possível visualizar, em ordem decrescente, os processos com maior utilização de CPU e de memória, e os processos de usuário com maior tempo de execução, respectivamente. A partir dos dados apresentados, é possível observar como os processos relacionados ao browser Firefox aparecem no topo da lista em todos os casos.



```
Guilherme@localhost:~$ ps -eo pid,cmd,%mem,%cpu --sort=-%cpu
```

PID	CMD	%MEM	%CPU
560960	/usr/lib/firefox-esr/firefo	13.6	33.2
561159	/usr/lib/firefox-esr/firefo	8.2	8.1
561404	/usr/lib/firefox-esr/firefo	8.8	6.8
561367	/usr/lib/firefox-esr/firefo	5.7	5.7
561488	/usr/bin/gnome-screenshot -	1.2	2.0
1579	/usr/bin/pulseaudio --daemo	0.3	0.8
561297	/usr/lib/firefox-esr/firefo	1.9	0.8
560695	evince /home/Guilherme/Down	1.9	0.6
560724	/usr/libexec/gnome-terminal	1.2	0.6
561053	/usr/lib/firefox-esr/firefo	3.0	0.6
561400	/usr/lib/firefox-esr/firefo	2.5	0.5
806	/usr/lib/xorg/Xorg :0 -seat	2.5	0.4
561245	/usr/lib/firefox-esr/firefo	3.6	0.4
561447	[kworker/0:3-events]	0.0	0.4
560693	[kworker/0:1-events]	0.0	0.3
561526	/usr/lib/firefox-esr/firefo	1.7	0.3
561529	/usr/lib/firefox-esr/firefo	1.7	0.3
1563	/lib/systemd/systemd --user	0.1	0.2
561160	/usr/lib/firefox-esr/firefo	3.0	0.2
1077	/usr/sbin/mysqld	1.4	0.1
1823	metacity	0.5	0.1
16797	file-roller /home/Guilherme	0.8	0.1

Figura 7 – Processos com maior utilização de CPU.

```

Guilherme@localhost: ~
Guilherme@localhost:~$ ps -eo pid,cmd,%mem,%cpu --sort=-%mem
  PID CMD                                %MEM %CPU
 560960 /usr/lib/firefox-esr/firefo 13.0 39.0
 561159 /usr/lib/firefox-esr/firefo  8.1 10.8
 561367 /usr/lib/firefox-esr/firefo  5.8  9.2
 561245 /usr/lib/firefox-esr/firefo  3.6  0.6
 561160 /usr/lib/firefox-esr/firefo  3.0  0.3
 561053 /usr/lib/firefox-esr/firefo  2.9  0.6
 561093 /usr/lib/firefox-esr/firefo  2.5  0.2
 561276 /usr/lib/firefox-esr/firefo  2.4  0.1
   806 /usr/lib/xorg/Xorg :0 -seat  2.0  0.4
560695 evince /home/Guilherme/Down  1.9  0.7
561297 /usr/lib/firefox-esr/firefo  1.9  1.2
561404 /usr/lib/firefox-esr/firefo  1.7  0.0
561448 /usr/lib/firefox-esr/firefo  1.7  0.0
561400 /usr/lib/firefox-esr/firefo  1.7  0.0
  1077 /usr/sbin/mysqld              1.4  0.1
560724 /usr/libexec/gnome-terminal  1.2  0.5
  1953 /usr/bin/gnome-software --g  1.1  0.0
  1760 /usr/bin/gnome-flashback      1.1  0.0
561309 /usr/lib/firefox-esr/firefo  1.1  0.0
561014 /usr/lib/firefox-esr/firefo  0.9  0.0
  1926 gnome-panel                  0.8  0.0
16797 file-roller /home/Guilherme  0.8  0.1

```

Figura 8 – Processos com maior utilização de memória.

```

Guilherme@localhost: ~
Guilherme@localhost:~$ ps -eo pid,user,cmd,time,%mem,%cpu --sort=-time
  PID USER      CMD                                TIME %MEM %CPU
 3233 Guilher+ /usr/lib/firefox-esr/firefo 00:01:42 9.7 36.3
 3471 Guilher+ /usr/lib/firefox-esr/firefo 00:00:16 7.7 5.7
 2175 root      /usr/libexec/packagekitd      00:00:13 1.0 3.5
 3528 Guilher+ /usr/lib/firefox-esr/firefo 00:00:13 5.1 5.0
   808 root      /usr/lib/xorg/Xorg :0 -seat 00:00:10 1.9 2.6
    1 root      /sbin/init splash             00:00:02 0.2 0.5
   629 root      /usr/lib/snapd/snapd          00:00:02 0.8 0.4
  1032 mysql    /usr/sbin/mysqld             00:00:02 10.5 0.7
  1783 Guilher+ /usr/libexec/ibus-extension  00:00:02 0.8 0.5
  1812 Guilher+ metacity                     00:00:02 0.8 0.5
  1916 Guilher+ gnome-panel                   00:00:02 1.8 0.7
 2992 Guilher+ /usr/libexec/gnome-terminal  00:00:02 1.2 0.8
   167 root      [kworker/0:2-events]         00:00:01 0.0 0.2
  1573 Guilher+ /usr/libexec/tracker-miner-  00:00:01 0.7 0.3
  1950 Guilher+ /usr/bin/gnome-software --g  00:00:01 2.1 0.3
    2 root      [kthreadd]                   00:00:00 0.0 0.0
    3 root      [rcu_gp]                     00:00:00 0.0 0.0
    4 root      [rcu_par_gp]                 00:00:00 0.0 0.0
    6 root      [kworker/0:0H-kblockd]       00:00:00 0.0 0.0
    7 root      [kworker/0:1-events]         00:00:00 0.0 0.1
    9 root      [mm_percpu_wq]               00:00:00 0.0 0.0
   10 root      [ksoftirqd/0]                00:00:00 0.0 0.0

```

Figura 9 – Processos com maior tempo de execução.





manipular e verificar os processos, especialmente os zumbis e, por fim, foi possível programar um código em C capaz de criar indefinidamente processos filhos e observar seus efeitos no sistema operacional.

## 6 Conclusões

Por fim, uma vez que os processos são uma parte essencial do funcionamento pleno de um sistema operacional, ao compreendê-los e ser capaz operá-los, é possível realizar diversas ações importantes em Linux.

## 7 Referências

TANENBAUM, A. S.; BOS, H. *Sistemas Operacionais Modernos*. [S.l.]: Pearson Education do Brasil, 2016. Citado na página [4](#).