

# SISTEMAS OPERACIONAIS

Marcos Bezner Rampaso  
2149435

Reginaldo Gregório de Souza Neto  
2252813

Carolina Yumi Fujii  
2335468

## **Laboratório 03: Manipulação de Threads**

Relatório técnico de atividade prática  
solicitado pelo professor Rodrigo  
Campiolo na disciplina de Sistemas  
Operacionais do Bacharelado em  
Ciência da Computação da  
Universidade Tecnológica Federal do  
Paraná.

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

Campo Mourão  
Agosto / 2022

---

## Sumário

<b>1 Introdução.....</b>	<b>3</b>
<b>2 Objetivos.....</b>	<b>3</b>
<b>3 Fundamentação.....</b>	<b>3</b>
<b>4 Materiais.....</b>	<b>3</b>
<b>5 Procedimentos.....</b>	<b>4</b>
<b>6 Conclusão .....</b>	<b>9</b>
<b>7 Referências .....</b>	<b>9</b>

# 1. Introdução

O relatório consiste em uma documentação que envolve a manipulação de threads no sistema operacional. Sendo dividido em duas etapas onde a primeira corresponde à manipulação de comando no terminal Linux a fim de checar o comportamento das threads no sistema. A segunda faz menção à parte de programação utilizando threads.

## 2. Objetivos

O objetivo do trabalho consiste em aprimorar os conhecimentos estudados em sala de aula sobre a funcionalidade de alguns comandos Linux para a manipulação de threads. Além de construir programas que desenvolvam determinadas soluções envolvendo threads.

## 3. Fundamentação

Segundo Maziero, uma thread é definida como sendo um fluxo de execução independente. Um processo pode conter uma ou mais threads, cada uma executando seu próprio código e compartilhando recursos com as demais threads localizadas no mesmo processo.

As perguntas foram disponibilizadas pelo Prof. Dr. Rodrigo Campiolo, a fim de que, através de suas aulas ministradas, bem como os materiais didáticos disponibilizados, seja possível assimilar e compreender o conceito e o funcionamento das threads.

## 4. Materiais

- Processador: AMD A8-5500B APU com Radeon(tm) HD Graphics
- RAM instalada: 8,00 GB (utilizável: 5,95 GB)
- Tipo de sistema: Sistema Operacional Mint

## 5. Procedimentos

Na primeira parte deste estudo sobre *threads*, o comando *htop* foi executado a fim de informar o número de *threads* atualmente em execução no sistema, o número foi de 537.

```

a2149435@dacom-labs: ~
Arquivo  Editar  Exibir  Pesquisar  Terminal  Ajuda

1  [  ]  5.7%  Tasks: 118, 537 thr: 1 running
2  [  ]  6.0%  Load average: 1.58 1.05 0.70
3  [  ]  7.0%  Uptime: 00:19:56
4  [  ]  8.7%
Mem [|||||] 2.28G/7.66G
Swp [|||||] 0K/5.83G

PID USER   PRI  NI  VIRT   RES   SHR  S  CPU%  MEM%   TIME+  Command
1865 a2149435 -6   0 1560M 19920 15220 S  3.9  0.2  0:52.46 /usr/bin/pulseaudio --daemonize=no --log-target=journal
1780 a2149435  9   0 1560M 19920 15220 S  3.9  0.2  0:52.99 /usr/bin/pulseaudio --daemonize=no --log-target=journal
839  root    20   0 672M  90M  6904 S  2.6  1.3  1:02.15 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -nov
2878 a2149435 20   0 491M  40312 34992 S  2.6  0.6  0:03.84 mate-terminal
3870 a2149435 20   0 1812  6096  832 R  2.0  0.1  0:00.10 htop
1087 a2149435 20   0 2290M 378M 36384 S  1.3  4.8  0:05.82 /usr/sbin/mysqld
740  a2149435 20   0 2290M 378M 36384 S  0.7  4.8  0:10.80 /usr/sbin/mysqld
2068 a2149435 20   0 734M  8556  4336 S  0.7  1.1  0:05.34 mintmenu
1  root    20   0 164M  11716  452 S  0.0  0.1  0:01.59 /sbin/init splash
322  root    19   0 60760 20368 21796 S  0.0  0.3  0:00.51 /lib/systemd/systemd-journald
353  root    20   0 24548  7564  4020 S  0.0  0.1  0:00.88 /lib/systemd/systemd-udevd
478  root    20   0 2532  768  704 S  0.0  0.0  0:00.35 /usr/sbin/acpid
482  messagebu 20   0 25004  852  4252 S  0.0  0.1  0:00.48 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --
591  root    20   0 412M  21116 20008 S  0.0  0.3  0:00.02 /usr/sbin/NetworkManager --no-daemon
608  root    20   0 412M  21116 20008 S  0.0  0.3  0:00.04 /usr/sbin/NetworkManager --no-daemon
483  root    20   0 412M  21116 20008 S  0.0  0.3  0:00.32 /usr/sbin/NetworkManager --no-daemon
532  root    20   0 81888  836  492 S  0.0  0.0  0:00.00 /usr/sbin/irqbalance --foreground
495  root    20   0 81888  836  492 S  0.0  0.0  0:00.10 /usr/sbin/irqbalance --foreground
506  root    20   0 79920 20772 11924 S  0.0  0.3  0:00.17 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
576  syslog   20   0 229M  4580  880 S  0.0  0.1  0:00.02 /usr/sbin/rsyslogd -n -iNONE
577  syslog   20   0 229M  4580  880 S  0.0  0.1  0:00.00 /usr/sbin/rsyslogd -n -iNONE
578  syslog   20   0 229M  4580  880 S  0.0  0.1  0:00.03 /usr/sbin/rsyslogd -n -iNONE
514  syslog   20   0 229M  4580  880 S  0.0  0.1  0:00.08 /usr/sbin/rsyslogd -n -iNONE
930  root    20   0 1070M 38336 19252 S  0.0  0.5  0:00.10 /usr/lib/snapd/snapd
931  root    20   0 1070M 38336 19252 S  0.0  0.5  0:00.00 /usr/lib/snapd/snapd
932  root    20   0 1070M 38336 19252 S  0.0  0.5  0:00.08 /usr/lib/snapd/snapd
933  root    20   0 1070M 38336 19252 S  0.0  0.5  0:00.05 /usr/lib/snapd/snapd

F1=help F2=setup F3=search F4=filter F5=tree F6=sortby F7=nice F8=nice F9=kill F10=quit

```

Figura 1 - utilizando o comando htop

Para checar o programa que mais possui threads, foi utilizado o comando *pstree*, onde através da visão das árvores, verifica-se qual programa possui mais processos filhos, e a partir disso, depreende-se qual o programa com mais *threads*, no caso do sistema utilizado, foi o processo *systemd* (Figura 2).

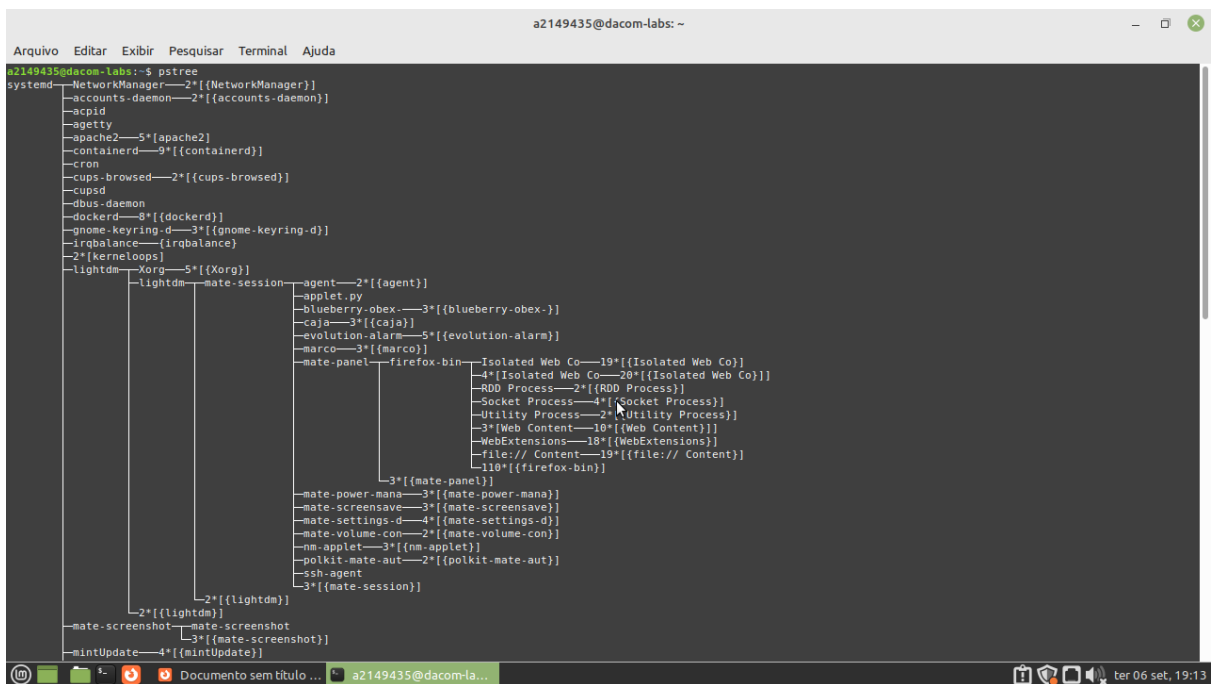


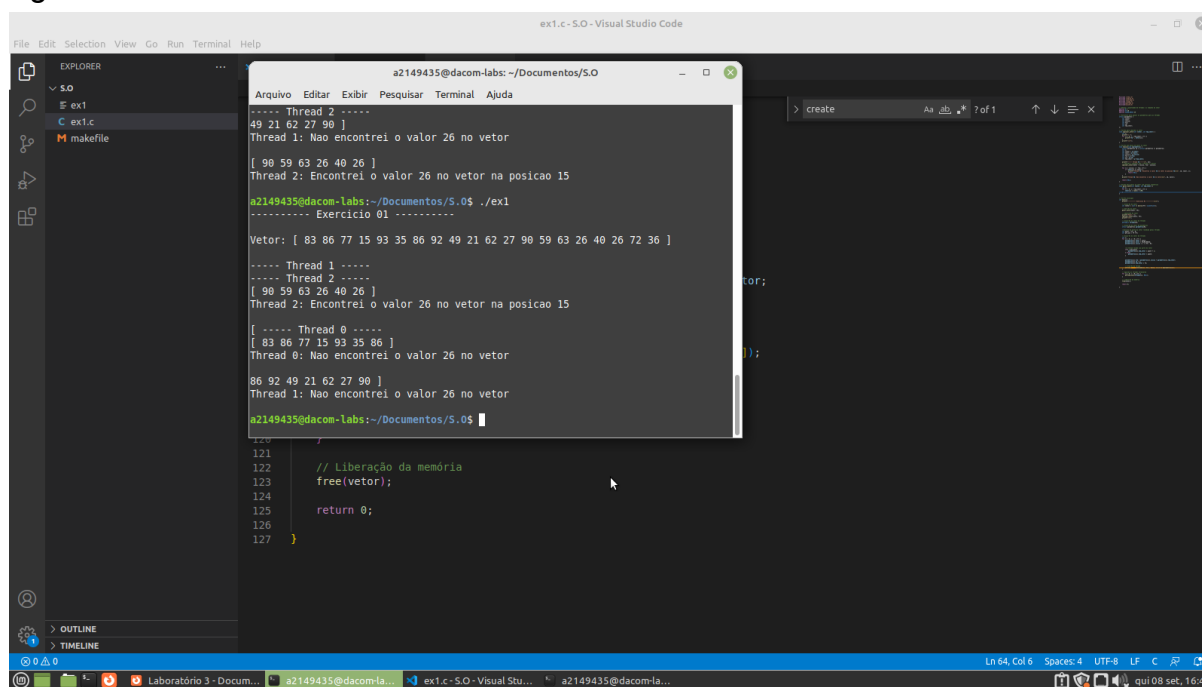
Figura 2 - Utilizando o comando pstree.

Para ser possível a checagem do número máximo de *threads* que o sistema suporta, é necessário utilizar o seguinte comando no terminal: `cat /proc/sys/kernel/threads-max`, que evidencia que o sistema pode ter até 61521 *threads* em execução (Figura 3).

```
a2149435@dacom-labs:~$ cat /proc/sys/kernel/threads-max
61521
a2149435@dacom-labs:~$
```

Figura 3 - Utilização do comando `cat /proc/sys/kernel/threads-max`.

Após a utilização do terminal para a visualização das *threads* do sistema, foram feitos 3 algoritmos diferentes, com base nos estudos acima. O primeiro deles foi um algoritmo de busca em vetor utilizando *threads* (Figura 4). Após o primeiro, foi feito também um algoritmo que utiliza duas *threads* para a pesquisa de elementos em um vetor que possua apenas os valores de 0 a 9 (Figura 5). E por fim, o último algoritmo criado foi um algoritmo que suporta *multithreading*, onde o mesmo calcula a mediana de uma matriz e devolve o resultado em um vetor **M**, também calcula a média aritmética de cada coluna. O algoritmo gera matrizes aleatórias de tamanhos escolhidos pelo usuário, as Figuras 6, 7 e 8 evidenciam o funcionamento do algoritmo.



```
Arquivo Editar Exibir Pesquisar Terminal Ajuda
a2149435@dacon-labs: ~/Documentos/S.O
----- Thread 2 -----
49 21 62 27 90 ]
Thread 1: Nao encontrei o valor 26 no vetor
[ 98 59 63 26 40 26 ]
Thread 2: Encontrei o valor 26 no vetor na posicao 15
a2149435@dacon-labs:~/Documentos/S.O$ ./ex1
----- Exercício 01 -----
Vetor: [ 83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36 ]
----- Thread 1 -----
----- Thread 2 -----
[ 98 59 63 26 40 26 ]
Thread 2: Encontrei o valor 26 no vetor na posicao 15
[ ----- Thread 0 -----
[ 83 86 77 15 93 35 86 ]
Thread 0: Nao encontrei o valor 26 no vetor
86 92 49 21 62 27 90 ]
Thread 1: Nao encontrei o valor 26 no vetor
a2149435@dacon-labs:~/Documentos/S.O$

120
121
122 // Liberação da memória
123 free(vetor);
124
125 return 0;
126
127 }
```

Figura 4 - Demonstração do algoritmo de busca utilizando *threads*.

```

a2149435@dacon-labs: ~/Documentos/S.O
Arquivo Editar Exibir Pesquisar Terminal Ajuda
a2149435@dacon-labs:~/Documentos/S.O$ gcc ex2.c -o ex2.exe -lpthread
a2149435@dacon-labs:~/Documentos/S.O$ ./ex2.exe
----- Exercício 02 -----

Vetor: [ 3 6 7 5 3 5 6 2 9 1 2 7 0 9 3 6 0 6 2 6 1 8 7 9 2 0 2 3 7 5 ]

----- Thread 1 -----
----- Thread 0 -----
[ 3 6 7 5 3 5 6 2 9 [ 1 2 6 7 0 0 9 6 2 3 6 ]
1 8 7 9 2 0 2 3 7 5 ]

Vetor de resposta: [ 3 2 5 4 0 3 5 4 1 3 ]

A frequência do numero 0 eh 3
A frequência do numero 1 eh 2
A frequência do numero 2 eh 5
A frequência do numero 3 eh 4
A frequência do numero 4 eh 0
A frequência do numero 5 eh 3
A frequência do numero 6 eh 5
A frequência do numero 7 eh 4
A frequência do numero 8 eh 1
A frequência do numero 9 eh 3
a2149435@dacon-labs:~/Documentos/S.O$

44
45 }
46
47 // Função que busca um valor no vetor
48 void *busca(void *parametros) {
49     struct parametros *p = (struct parametros *) parametros;
50     int i;
51     int *vetor = p->vetor;
52     int *vetor_resposta = p->vetor_resposta;
53     int inicio = p->inicio;
54     int fim = p->fim;
55     int id = p->id;
56     int tam_vetor = p->tam_vetor;
57
58     printf("----- Thread %d ----- \n", id);

```

Figura 5 - Demonstração do algoritmo que busca frequência de números de 0 a 9 utilizando *threads*.

```

a2149435@dacon-labs: ~/Documentos/S.O
Arquivo Editar Exibir Pesquisar Terminal Ajuda
61 59 55 62 2 51 47 22 79 41 15 60 35 98 15 58 28 15 25 48
38 99 42 43 2 66 68 39 9 67 98 68 15 53 30 14 3 77 35 79
16 48 37 48 46 49 7 71 63 31 18 2 28 59 44 30 23 10 66 29
75 64 96 87 15 25 2 17 2 34 94 15 81 31 63 26 80 67 96 41
95 12 42 24 70 84 53 90 91 17 20 64 80 14 52 94 38 51 12 37
84 4 52 64 34 13 89 12 79 83 53 74 95 92 95 63 74 48 54 66
64 71 38 42 84 79 37 21 38 46 55 15 49 5 78 89 17 65 92 93
46 45 66 41 37 61 3 12 8 56 75 69 25 3 12 7 82 48 27 10
91 80 24 38 84 99 18 99 62 10 92 9 52 58 47 89 18 49 98 25
5 73 93 27 73 5 34 55 50 58 65 39 38 86 76 23 84 94 20 44
2 10 52 54 68 99 41 85 46 39 8 49 10 1 75 82 6 9 36 53
67 98 92 5 82 68 25 64 60 44 9 62 54 58 14 20 55 54 3 2
90 10 50 90 10 25 79 13 32 13 66 98 11 58 1 93 24 26 55 84
67 63 46 19 20 59 36 72 11 36 73 1 43 21 97 51 46 74 63 75
87 27 73 97 82 73 88 7 96 44 90 62 5 34 78 22 90 13 94 98
48 67 98 91 86 93 42 30 67 3 5 54 30 77 50 10 48 38 16 43
79 4 3 82 37 80 4 25 99 97 23 39 63 19 28 49 12 69 78 77

----- MEDIANA -----
Thread 0: 200 Linhas
Thread 1: 200 Linhas
Thread 2: 200 Linhas
Thread 3: 200 Linhas
Thread 4: 200 Linhas

```

Figura 6 - Divisão das medianas para cada *thread* individual

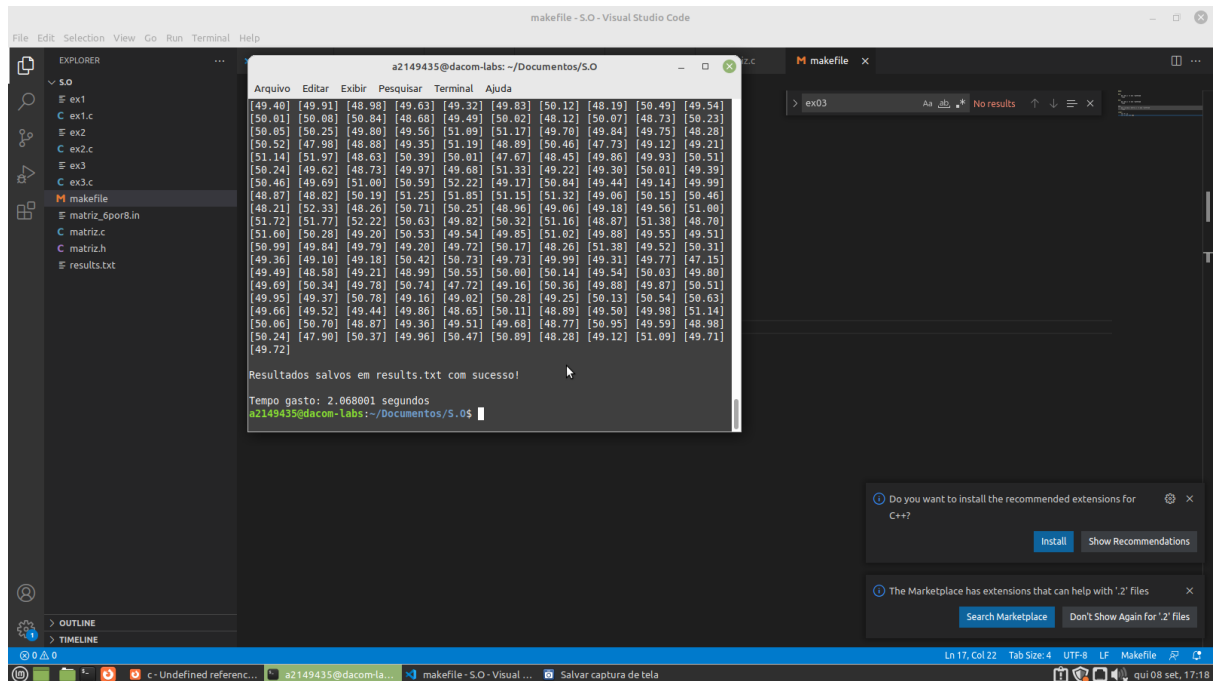


Figura 7 - Tempo de execução do algoritmo em uma matriz 1000 x 1000 utilizando 8 threads

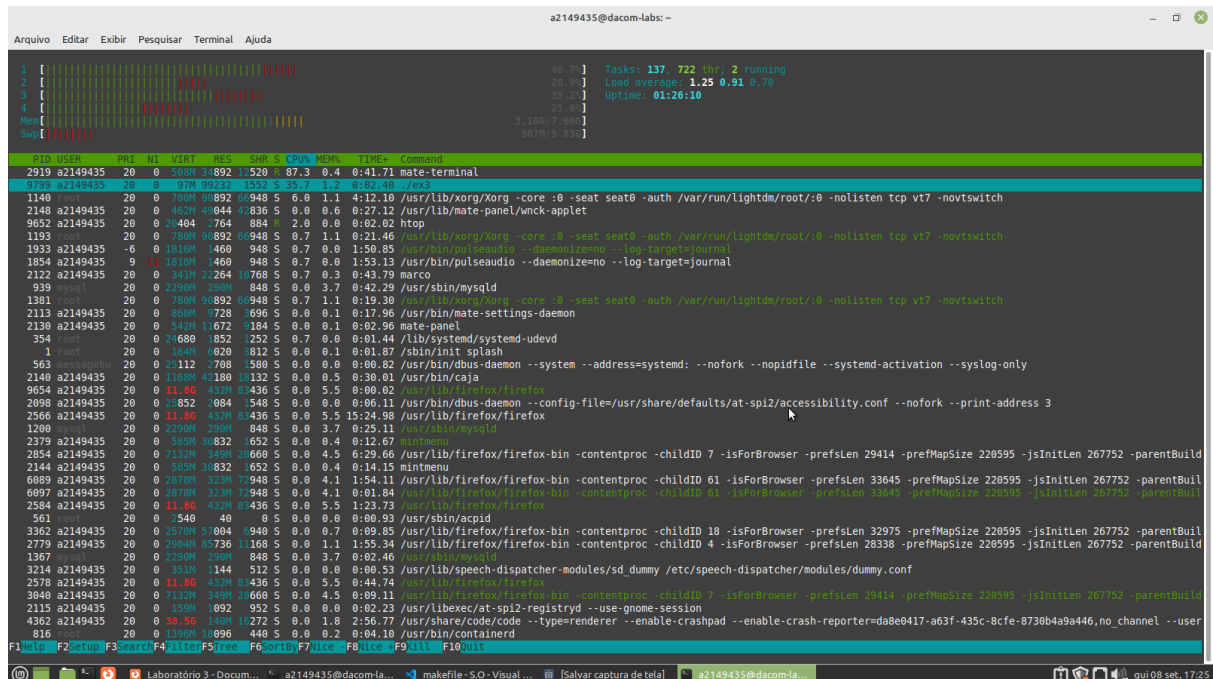


Figura 8 - Utilização do Hardware utilizando uma matriz de tamanho 1000 x 1000 utilizando 6 threads



```

a2149435@dacon-labs: ~/Documentos/S.O
Arquivo  Editar  Exibir  Pesquisar  Terminal  Ajuda
75 14 65 92 13 73 61 64 16 44 5 56 89 90 74 90 49 61 27 60
16 83 41 79 70 14 27 43 13 57 49 85 69 14 77 81 84 37 44 1
80 48 54 68 36 28 58 84 88 83 43 3 66 83 81 35 96 9 77 7
63 24 92 32 36 69 12 20 6 55 18 84 1 72 50 36 97 8 19 86
88 61 86 53 44 67 87 38 73 62 43 37 84 35 66 20 2 75 37 6
30 55 89 38 25 39 64 22 44 82 6 31 43 91 83 84 59 68 20 30
28 62 64 12 96 31 29 98 6 66 2 34 19 90 61 41 27 25 62 68
7 66 98 47 57 79 29 14 47 49 44 75 9 6 84 6 36 14 2 40
77 3 71 95 90 32 36 15 55 95 82 59 61 79 7 17 58 33 28 4
81 69 78 90 75 62 93 9 73 92 46 50 92 17 45 80 47 78 94 2
72 74 68 33 51 64 47 10 97 73 13 76 42 88 64 15 48 55 24 22

----- MEDIANA -----
Thread 0: 4 linhas
Thread 1: 4 linhas
Thread 2: 4 linhas
Thread 3: 4 linhas
Thread 4: 4 linhas
1 2 4 7 16 17 19 22 34 34 50 50 57 63 67 68 69 82 97 98
1 2 10 12 14 15 17 25 33 34 35 37 40 42 45 55 63 69 79 85
1 17 19 22 33 34 43 49 54 61 63 78 79 81 82 87 88 91 93 95
4 6 8 12 13 15 24 26 27 34 38 42 48 51 55 70 71 75 90 91
4 5 13 14 14 22 22 26 26 33 37 40 51 54 60 63 85 86 88 91
2 3 9 16 29 30 31 43 45 54 54 60 75 82 82 84 87 93 94 97
16 16 23 29 36 38 40 45 48 53 53 56 65 68 71 74 79 92 97
8 11 19 26 31 33 42 45 52 60 69 69 79 83 83 84 85 88 95 99
2 6 14 18 31 37 41 48 48 55 59 60 66 75 75 80 86 87 89 98
5 13 14 16 27 44 49 56 60 61 61 64 65 73 74 75 89 90 90 92
1 13 14 14 16 27 37 41 43 44 49 57 69 70 77 79 81 83 84 85
3 7 9 28 35 36 43 48 54 58 66 68 77 80 81 83 83 84 88 96
1 6 8 12 18 19 20 24 22 36 36 50 55 63 69 72 84 86 92 97
2 6 20 35 37 37 38 43 44 53 61 62 66 67 73 75 84 86 87 88
6 20 22 25 30 30 30 31 39 43 44 55 59 64 68 82 83 84 89 91
2 6 12 19 25 27 28 29 31 34 41 61 62 62 64 66 68 90 96 98
2 6 6 7 9 14 14 29 36 40 44 47 47 49 57 66 75 79 84 98
3 4 7 15 17 28 32 33 36 55 58 59 61 71 77 79 82 90 95 95
2 9 17 45 46 47 50 62 69 73 75 78 80 81 90 92 92 93 94
10 13 15 22 24 33 42 47 48 51 55 60 64 64 72 73 74 76 88 97

Mediana: [50] [35] [63] [38] [37] [54] [53] [69] [59] [61] [49] [66] [36] [61] [44] [41] [44] [58] [75] [55]
Media: [48.25] [47.70] [64.10] [51.20] [52.45] [55.60] [49.35] [46.55] [49.00] [50.70] [44.20] [51.20] [50.50] [57.65] [58.20] [52.60] [43.20] [46.15] [39.00] [30.95]
Resultados salvos em results.txt com sucesso!
Tempo gasto: 0.002693 segundos
a2149435@dacon-labs: ~/Documentos/S.O$

```

Figura 8 - Tempo de execução de uma matriz de tamanho 20 x 20 utilizando 4 threads.

## 6. Conclusão

Após realizados todos os procedimentos solicitados na parte 1 e 2 dessa atividade, dá-se por concluído este laboratório que serviu como meio de aprendizagem sobre threads em sistemas operacionais. Depreende-se que a utilização de mais threads em matrizes maiores, diminuiu o tempo de execução, enquanto o uso de muitas threads em uma pequena matriz, acaba por fazer o algoritmo demorar um tempo maior para execução.

Por fim, ressaltamos que todos os códigos seguem em anexo junto ao relatório.

## 7. Referências

- <https://moodle.utfpr.edu.br/course/view.php?id=2593>
- <https://www.baeldung.com/linux/max-threads-per-process>
- <http://wiki.inf.ufpr.br/maziero/lib/exe/fetch.php?media=socm:socm-livro.pdf>