

## Disciplina: Sistemas Operacionais I

### Exercício 3

Baseando se nos slides da Aula 11 faça:

1. Elabore um programa ilustrando o uso de threads no Linux com a biblioteca PThreads.
  - a. Explique o código fonte utilizado

```

C thread.c x
ex3 > C thread.c > ...
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <pthread.h>
4
5  /*Rotina que será executada*/
6  void * routine(void *arg);
7
8  int main (int argc, char *argv[])
9  {
10     pthread_t thread_id;
11     void * thread_res;
12     int rstatus;
13
14     /*tenta iniciar o thread, indicando a função 'routine' e passando como argumento a string
15     "Minha primeira Thread!"/
16     rstatus = pthread_create (&thread_id, NULL, routine, (void*)"Minha primeira Thread!");
17
18     /*verificar se ocorreu algum erro na chamada de pthread_create*/
19     if(rstatus != 0)
20     {
21         printf ("Erro ao criar o thread.\n");
22         exit(EXIT_FAILURE);
23     }
24
25     printf ("Thread criado com sucesso!\n");
26
27     /*aguarda finalização do thread identificado por thread_id. O retorno é passado pelo ponteiro
28     thread_res*/
29     rstatus = pthread_join (thread_id, &thread_res);
30
31     /*verificar se ocorreu algum erro na chamada de pthread_join*/
32     if(rstatus != 0)
33     {
34         printf ("Erro ao aguardar finalização do thread.\n");
35         exit(EXIT_FAILURE);
36     }
37
38     /*exibe o valor de retorno da função 'routine'*/
39     printf ("Thread finalizado! Retorno = %s\n", (char *)thread_res);
40
41     return 0;
42 }
43
44 void * routine(void *arg)
45 {
46     /*exibe o argumento recebido*/
47     printf("Argumento: %s\n", (char *)arg);
48
49     /*finaliza a função retornando o argumento que foi recebido*/
50     pthread_exit(arg);
51 }
52

```

b. Descreva os resultados obtidos

```
Terminal - ariana@arcursino: ~/projetos/FATEC/so/ex3
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
ariana@arcursino:~/projetos/FATEC/so/ex3$ gcc -o thread thread.c -lpthread
ariana@arcursino:~/projetos/FATEC/so/ex3$ ./thread
Thread criado com sucesso!
Argumento: Minha primeira Thread!
Thread finalizado! Retorno = Minha primeira Thread!
ariana@arcursino:~/projetos/FATEC/so/ex3$
```

2. Elabore um programa ilustrando a relação Produtor-Consumidor por thread em Java.

a. Explique o código fonte utilizado

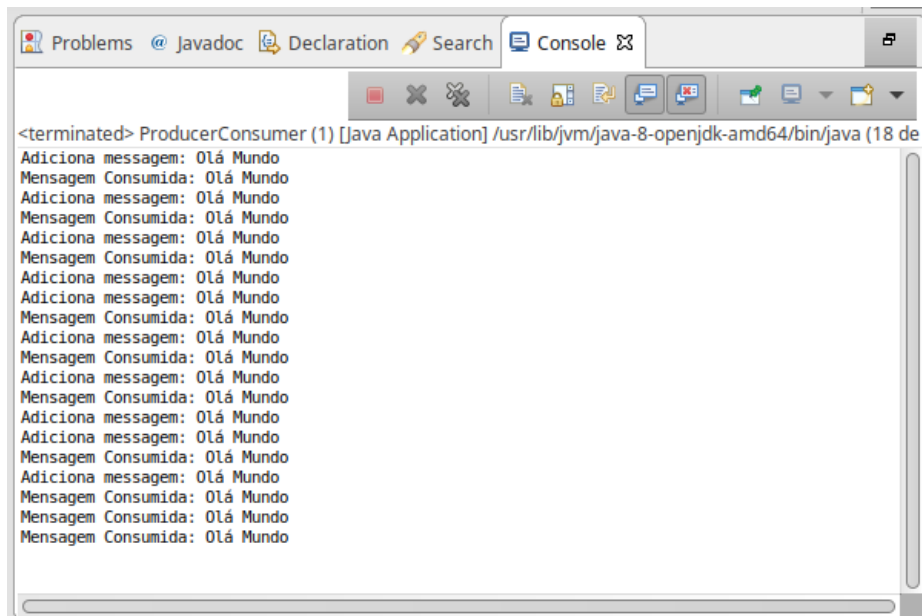
```
1 package producerconsumer.main;
2
3 import static java.lang.System.out;
4
5
6 /**
7  * Começa 2 threads, um produtor e um consumidor
8  * <br>
9  * Ambas as threads compartilham o mesmo BlockingQueue
10  */
11
12 public class ProducerConsumer {
13
14     private static final int MSG_NBR = 10;
15
16     private final BlockingQueue<String> queue = new ArrayBlockingQueue<>(10);
17
18     public static void main(String[] args) {
19         new ProducerConsumer().startEngine();
20     }
21
22     public void startEngine() {
23         startProducer();
24         startConsumer();
25     }
26
27     // thread Produtor
28     private void startProducer() {
29
30         final MyProducer<String> myProducer = new MyProducer<>(queue);
31         final Supplier<String> supplier = () -> "Olá Mundo";
32         new Thread(() -> {
33             for (int i = 0; i < MSG_NBR; i++) {
34                 myProducer.produce(supplier);
35             }
36         }).start();
37     }
38
39     // thread Consumidor
40     private void startConsumer() {
41
42         final MyConsumer<String> myConsumer = new MyConsumer<>(queue);
43         final Consumer<String> consumer = () -> out.println("Mensagem Consumida: " + s);
44         new Thread(() -> {
45             for (int i = 0; i < MSG_NBR; i++)
46                 myConsumer.consume(consumer);
47         }).start();
48     }
49
50     static class MyProducer<T> {
51
52         private BlockingQueue<T> queue;
53
54         public MyProducer(BlockingQueue<T> queue) {
55             this.queue = queue;
56         }
57
58         /**
59          * Insere o objeto "supplier" na fila
60          *
61          * @param supplier
62          *     É responsável por fornecer o objeto que será colocado
63          *     na fila
64          */
65         public void produce(Supplier<T> supplier) {
66             final T msg = supplier.get();
67             try {
68                 queue.put(msg);
69                 out.println("Adiciona mensagem: " + msg);
70
71                 // Simula o processo running
72                 Thread.sleep(900);
73             } catch (InterruptedException e) {
74                 throw new RuntimeException(e);
75             }
76         }
77     }
78
79     static class MyConsumer<T> {
80
81         private BlockingQueue<T> queue;
82
83         public MyConsumer(BlockingQueue<T> queue) {
84             this.queue = queue;
85         }
86
87         /**
88          * Remove o objeto da fila
89          *
90          * @param consumer
91          *     É responsável por consumir o objeto que será retirado
92          *     da fila
93          */
94         public void consume(Consumer<T> consumer) {
95             try {
96                 T msg = queue.take();
97                 consumer.accept(msg);
98             } catch (InterruptedException e) {
99                 throw new RuntimeException(e);
100             }
101         }
102     }
103 }
```

```

84
85 static class MyConsumer<T> {
86
87     private BlockingQueue<T> queue;
88
89     public MyConsumer(BlockingQueue<T> queue) {
90         this.queue = queue;
91     }
92
93     /**
94      * Recupera um objeto do início da fila e passa-o para o
95      * consumidor
96      * @param consumer
97      * Contém a lógica sobre o que fazer com o objeto recuperado
98      */
99
100    public void consume(Consumer<T> consumer) {
101        try {
102            consumer.accept(queue.take());
103
104            // Simula o processo running
105            MILLISECONDS.sleep(1250);
106
107        } catch (InterruptedException e) {
108            throw new RuntimeException(e);
109        }
110    }
111 }
112 }

```

b. Descreva os resultados obtidos



```

<terminated> ProducerConsumer (1) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (18 de
Adiciona mensagem: Olá Mundo
Mensagem Consumida: Olá Mundo
Adiciona mensagem: Olá Mundo
Mensagem Consumida: Olá Mundo
Adiciona mensagem: Olá Mundo
Mensagem Consumida: Olá Mundo
Adiciona mensagem: Olá Mundo
Adiciona mensagem: Olá Mundo
Mensagem Consumida: Olá Mundo
Adiciona mensagem: Olá Mundo
Mensagem Consumida: Olá Mundo
Adiciona mensagem: Olá Mundo
Adiciona mensagem: Olá Mundo
Mensagem Consumida: Olá Mundo
Adiciona mensagem: Olá Mundo
Mensagem Consumida: Olá Mundo
Adiciona mensagem: Olá Mundo
Mensagem Consumida: Olá Mundo

```

3. Elabore três processos (1x servidor com thread, 2x clientes) que se comunicam via socket TCP, observe que o servidor deverá aceitar mais de uma conexão.
- a. Explique o código fonte utilizado
- Servidor:

```

servidor_thread.py x cliente.py
ex3 > servidor_thread.py
1  #Servidor TCP
2  import socket
3  from threading import Thread
4
5  def conexao(con,cli):
6      while True:
7          msg = con.recv(1024)
8          if not msg: break
9          print (msg)
10         print ('Finalizando conexao do cliente', cli)
11         con.close()
12
13     # Endereco IP do Servidor
14     HOST = '127.0.0.1'
15     # Porta que o Servidor vai escutar
16     PORT = 5002
17     tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
18     orig = (HOST, PORT)
19     tcp.bind(orig)
20     tcp.listen(1)
21     while True:
22         con, cliente = tcp.accept()
23         print ('Concetado por ', cliente)
24         t = Thread(target=conexao, args=(con,cliente,))
25         t.start()

```

- Cliente:

```

servidor_thread.py x cliente.py x
ex3 > cliente.py
1
2  #Cliente TCP
3  import socket
4  # Endereco IP do Servidor
5  SERVER = '127.0.0.1'
6  # Porta que o Servidor esta escutando
7  PORT = 5002
8  tcp = socket.socket(socket.AF_INET,
9  socket.SOCK_STREAM)
10  dest = (SERVER, PORT)
11  tcp.connect(dest)
12  print ('Para sair use CTRL+X\n')
13  msg = input()
14  while msg != '\x18':
15      tcp.send (msg.encode())
16      msg = input()
17  tcp.close()

```

b. Descreva os resultados obtidos

```
Terminal - ariana@arcursino: ~/projetos/FATEC/so/ex3
Arquivo Editar Ver Terminal Abas Ajuda
ariana@arcursino:~$ cd projetos/FATEC/so/ex3/
ariana@arcursino:~/projetos/FATEC/so/ex3$ python3 cliente.py
Para sair use CTRL+X

Olá tudo bem? Eu sou o cliente 1
ah... agora entendi... sem acentos... cliente 1
Muito obrigada cliente 2. Foi de muita ajuda a sua explanacao
boa noite... cliente 2
boa noite servidor
^X
ariana@arcursino:~/projetos/FATEC/so/ex3$

Terminal - ariana@arcursino: ~/projetos/FATEC/so/ex3
Arquivo Editar Ver Terminal Abas Ajuda
ariana@arcursino:~$ cd projetos/FATEC/so/ex3/
ariana@arcursino:~/projetos/FATEC/so/ex3$ python3 cliente.py
Para sair use CTRL+X

Ola tudo bem? sem acento para nao cagar a mensagem... Eu sou o cliente 2
Denada cliente 1... Estou aqui para isso
boa noite cliente 1
boa noite servidor
^X
ariana@arcursino:~/projetos/FATEC/so/ex3$

21 while True:
22     con, cliente =
23     print ('Concetado por ('127.0.0.1', 33426)
24     t = Thread(target=
25     t.start()

Concetado por ('127.0.0.1', 33426)
Concetado por ('127.0.0.1', 33428)
b'Ol\xcc\xal tudo bem? Eu sou o cliente 1'
b'Ola tudo bem? sem acento para nao cagar a mensagem... Eu sou o cliente 2'
b'ah... agora entendi... sem acentos... cliente 1'
b'Muito obrigada cliente 2. Foi de muita ajuda a sua explanacao'
b'Denada cliente 1... Estou aqui para isso'
b'boa noite... cliente 2'
b'boa noite servidor'
b'boa noite cliente 1'
b'boa noite servidor'
Finalizando conexao do cliente ('127.0.0.1', 33426)
Finalizando conexao do cliente ('127.0.0.1', 33428)
```