

**UNIVERSIDADE ESTADUAL DE MATO GROSSO DO SUL UNIDADE
UNIVERSITÁRIA DE DOURADOS
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO DISCIPLINA DE
REDES DE COMPUTADORES**

GUSTAVO KERMAUNAR VOLOBUEFF

**IMPLEMENTAÇÃO DE UM SISTEMA ESCALONADOR
PROCESSADOR NO MODELO CLIENTE-SERVIDOR**

Dourados, MS

2024

GUSTAVO KERMAUNAR VOLOBUEFF

**IMPLEMENTAÇÃO DE UM SISTEMA ESCALONADOR
PROCESSADOR NO MODELO CLIENTE-SERVIDOR**

Trabalho acadêmico do curso Ciência da Computação
apresentado à disciplina de Redes de Computadores da
Universidade Estadual de Mato Grosso do Sul, Unidade
Universitária de Dourados como exigência para
composição da nota bimestral.

Prof. Dr. Rubens Barbosa Filho

Dourados, MS

2024

SUMÁRIO

1 INTRODUÇÃO.....	3
2 CLIENTE.CPP.....	4
3 PORTAL.CPP.....	5
4 SERVIDOR.CPP.....	6
5 BIBLIOTECAS CRIADAS.....	7
5.1 clienteUteis.h.....	8
5.2 EnderecoHandler.cpp.....	8
6 TESTES DE DEMONSTRAÇÃO.....	8
6.1 Utilizando Escalonamento Round-Robin.....	9
6.2 Utilizando Escalonamento Aleatório.....	15
7 TESTES CONFORME ESPECIFICAÇÃO.....	21
8 CONCLUSÃO.....	22
REFERÊNCIAS.....	23

1 INTRODUÇÃO

Para o desenvolvimento deste trabalho, foi utilizado a linguagem *C++* por conta da maior facilidade na manipulação de strings, porém, as bibliotecas utilizadas foram as mesmas no qual o professor Rubens utilizou em exemplos em sala de aula, que são da linguagem *C*.

Na questão de multi-conexões, foram utilizadas *threads*, tanto no portal quanto no servidor.

Para transmitir as mensagens, do cliente ao portal foi transmitido arquivo fonte por arquivo, um de cada vez, no qual o portal escalona um de cada vez também para cada servidor, e retornando para o cliente, que reinicia o processo.

2 CLIENTE.CPP

Sobre o programa cliente.cpp, é criado o *socket* e logo após é chamado a função *setsockopt(5)*, que auxilia na questão de conexão e reconexão imediatas dos programas, evitando erros e *bugs*.

Logo após, caso o parâmetro seja o nome da máquina, há um conversor na biblioteca criada, a *clienteUteis.h*, que será explicada na seção 5, no qual converte para o IP correspondente.

Então, com auxílio da biblioteca também criada *EnderecoHandler.h*, que também será explicada na seção 5, é *setado* o endereço para o *portal.cpp* e *bindado* com o *socket*.

A conexão é feita com o portal e ela fica presente durante todo o próximo laço.

Após isso há um laço infinito para que seja recebido e enviado os arquivos, onde, primeiramente o comando recebido é separado com a função *separa_string(1)*, e então caso a instrução seja L, ele lista os arquivos disponíveis para envio, no qual estão na pasta */arquivos_fonte/* no arquivo, na função *le_diretorio_funcao_L()*. Caso seja o comando S, ele recebe o comando, é separado em *tokens* cada argumento, o arquivo *.cpp* é aberto, inserido caracter a caracter dentro da variável *arq_fonte*.

Os arquivos são enviados um por um ao portal.

3 PORTAL.CPP

Sobre o *portal.cpp*, assim como no *cliente.cpp* é *setado* e *bindado* os *sockets* e o endereço, com o *cliente.cpp*, é o *listen(2)* é chamado, com o segundo parâmetro sendo 2 para receber 2 clientes. As conexões são aceitas e é criado 2 *threads*, uma para cada cliente, no qual é chamado a função *recebe_arquivos_fonte(1)* para manipular os envios e recebimentos de arquivos.

Dentro dessa função, são criados 3 endereços para os 3 servidores disponíveis, no qual possuem IP's diferentes e portas iguais, nos quais podem ser alterados nas constantes no início do código fonte. Após isso é feito o processo padrão antes explicado para *bindar* os endereços com os 3 *sockets* também criados, e após isso é feito a conexão imediata no laço com os 3 servidores, ficando assim sempre conectados.

Com isso, a *thread* fica no laço recebendo os arquivos do cliente, e a variável global *formaEscalonamento* salva qual será a forma de escalonar, caso seja Round-Robin, com o parâmetro *rr*, é enviado para os servidores na sequência com o contador *i*, sendo ele global para que seja compartilhada com ambas *threads* de clientes.

Para que não haja problemas de *deadlock* com as *threads*, o *mutex* é travado com o comando *m.lock()* e logo após executar, é dado *m.unlock()*.

No caso de escalonamento aleatório, com o parâmetro *altr*, ele irá sortear um valor de 0 a 2 para que seja escolhido um dos servidores na variável *i*, e da mesma forma é travado e destravado o *mutex*.

Todo processo de envio é sequencial, onde cada arquivo enviado espera o recebimento por parte do servidor e após isso devolvido ao cliente, podendo ser novamente enviado outro arquivo ao servidor.

4 SERVIDOR.CPP

No *servidor.cpp*, é feito o mesmo processo inicial que no *cliente.cpp* e *portal.cpp*, também criando uma thread para que seja manipulado o recebimento de arquivos, na função *compila_arquivos_fonte(1)*.

Nesta função, é criado um laço para o recebimento de arquivos do portal, onde, é convertido de *char** para *string*, e a função *coloca_em_arquivo(2)* é chamada, onde, ela irá receber o arquivo que está com o primeiro *token*, até o primeiro espaço em branco com o nome do arquivo a ser compilado, e um arquivo é criado com este nome, o nome, após isso, é removido da *string* recebida e o restante, no caso o código fonte, é inserido no arquivo criado.

Após isso é concatenado o comando para compilação, na variável *comandoComp*, que segue o padrão **g++ <nome do arquivo> 2>> erro.txt**. Este *erro.txt* irá armazenar o retorno da compilação, caso haja erros ou warnings na compilação. E então, o comando *system(1)* é chamado, com parâmetro a variável *comandoComp*, e o arquivo *erro.txt* é aberto, verificando caso haja algo dentro dele, caso haja, a função *le_resultado_insere_array(2)* é chamado, com o primeiro parâmetro sendo o valor 0, *que indica que é um erro*, e é inserido na *string resultado*, e caso a compilação tenha ocorrido corretamente, a função *le_resultado_insere_array(2)* é chamada, com o primeiro parâmetro sendo o valor 1, que indica que a compilação ocorreu corretamente, onde o arquivo *./a.out* é aberto e é inserido o resultado lá contido dentro da *string resultado*.

E então, é enviado ao *portal.cpp* o resultado da compilação.

5 BIBLIOTECAS CRIADAS

Para facilitar e limpar o código, foram criadas duas bibliotecas auxiliares, que estão dentro do diretório *includes* junto aos códigos fontes principais. No qual serão definidas a seguir.

5.1 clienteUteis.h

Nessa biblioteca, temos a função *hostname_para_ip(2)* que recebe o nome da máquina passada ao *cliente.cpp*, e é convertido para IP, basicamente utilizando as *structs* *hostent* e *in_addr*, no qual é passado à variável criada *he*, da struct *hostent*, o comando *gethostbyname(1)*, no qual é passado para *lista_addr*, da *struct in_addr*, a primeira ocasião deste nome de máquina encontrada e então é copiado para a variável *IP*, o IP encontrado.

Esta função foi retirada do *website* https://www.binarytides.com/hostname-to-ip-address-c-sockets-linux/#google_vignette como inspiração.

Há também a função *separa_string(1)* que recebe como parâmetro a *string* do comando do cliente, e ela separa o comando pelos delimitadores “[“, “,” e “]”, retornando a string sem estes delimitadores. Este algoritmo foi retirado de inspiração do website <https://www.geeksforgeeks.org/how-to-split-string-by-multiple-delimiters-in-cpp/>.

E, por fim, temos a função *le_diretorio_funcao_L()*, que lista dentro do diretório */arquivos_fonte/* os arquivos disponíveis para enviar ao portal.

5.2 EnderecoHandler.cpp

Nesta biblioteca, é criada a classe *EnderecoHandler*, no qual irá nos auxiliar com a conexão entre cliente-portal-servidor, onde só foi transferido para classes o que já possuíamos, com o único atributo sendo *struct sockaddr_in addr*.

Esta classe possui 2 construtores, um deles sendo para conexão (o que possui parâmetro *ip*) e o de receber conexões (o que possui parâmetro tipo).

6 TESTES DE DEMONSTRAÇÃO

Seguem alguns *printscreens* do funcionamento do sistema nos computadores do laboratório 4. No caso dos testes, deixei a impressão nos servidores e no portal para ser verificado caso os arquivos estejam sendo entregues e distribuídos da forma correta, no arquivo enviado NÃO terá estas impressões, somente no cliente.

Irei apresentar dois casos de teste para cada algoritmo de escalonamento, um deles sendo utilizando um só cliente, outro sendo dois clientes simultâneos.

6.1 Utilizando Escalonamento Round-Robin

Segue o primeiro teste com um só cliente:

```
rgm47006@l4m09u:~/Documentos/Trabalho_1$ ./cliente 172.26.4.127
rgm47006@l4m09u:~/Documentos/Trabalho_1$ ./cliente 172.26.4.127 47006
Conectado.
S sqrt.cpp ola_usuario.cpp
5
Ola usuario! Como esta?
```

Cliente 1

```
./porrgm47006@l4m08u:~/Documentos/Trabalho_1$ ./portal rr
Conectado
Conectado
Conectado
Conectado
Conectado
Conectado
sqrt.cpp #include <iostream>
#include <cmath>

int main(){
    std::cout << sqrt(25) << std::endl;

    return 0;
}
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
```

Portal

```
rgm47006@l4m04u:~/Documentos/Trabalho_1$ ./servidor
sqrt.cpp #include <iostream>
#include <cmath>

int main(){
    std::cout << sqrt(25) << std::endl;

    return 0;
}
```

Servidor 1

```
rgm47006@l4m21u:~/Documentos/Trabalho_1$ ./servidor  
ola_usuario.cpp #include <iostream>  
  
int main(){  
    std::cout << "Ola usuario! Como esta?" << std::endl;  
    return 0;  
}
```

Servidor 2

```
rgm47006@l4m22u:~/Documentos/Trabalho_1$ ./servidor
```

Servidor 3

E aqui segue o teste utilizando dois clientes simultâneos:

```
rgm47006@l4m09u:~/Documentos/Trabalho_1$ ./cliente 172.26.4.127 47006
Conectado.
S sqrt.cpp com_erro.cpp ola_usuario.cpp
5

com_erro.cpp: In function 'int main()':
com_erro.cpp:2:10: error: 'cout' is not a member of 'std'
  2 |         std::cout << "Era pra imprimir né" << std::endl;
    |         ^~~~~
com_erro.cpp:1:1: note: 'std::cout' is defined in header '<iostream>'; did you forget to '#include <iostream>'?
+++ |+#include <iostream>
  1 | int main(){
com_erro.cpp:2:48: error: 'endl' is not a member of 'std'
  2 |         std::cout << "Era pra imprimir né" << std::endl;
    |                                                ^~~~~
com_erro.cpp:1:1: note: 'std::endl' is defined in header '<ostream>'; did you forget to '#include <ostream>'?
+++ |+#include <ostream>
  1 | int main(){

Ola usuario! Como esta?
```

Cliente 1

```
rgm47006@l4m10u:~/Documentos/Trabalho_1$ ./cliente 172.26.4.127 47006
Conectado.
S ola_usuario.cpp com_erro.cpp
Ola usuario! Como esta?

com_erro.cpp: In function 'int main()':
com_erro.cpp:2:10: error: 'cout' is not a member of 'std'
  2 |         std::cout << "Era pra imprimir né" << std::endl;
    |         ^~~~~
com_erro.cpp:1:1: note: 'std::cout' is defined in header '<iostream>'; did you forget to '#include <iostream>'?
+++ |+#include <iostream>
  1 | int main(){
com_erro.cpp:2:48: error: 'endl' is not a member of 'std'
  2 |         std::cout << "Era pra imprimir né" << std::endl;
    |                                                ^~~~~
com_erro.cpp:1:1: note: 'std::endl' is defined in header '<ostream>'; did you forget to '#include <ostream>'?
+++ |+#include <ostream>
  1 | int main(){
```

Cliente 2

```
rgm47006@l4m08u: ~/Documentos/Trabalho_1
rgm47006@l4m08u:~/Documentos/Trabalho_1$ ./portal rr
Conectado
Conectado
Conectado
Conectado
Conectado
Conectado
Conectado
sqrt.cpp #include <iostream>
#include <cmath>

int main(){
    std::cout << sqrt(25) << std::endl;
    return 0;
}
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
com_erro.cpp int main(){
    std::cout << "Era pra imprimir né" << std::endl;
    return 0;
}
com_erro.cpp int main(){
    std::cout << "Era pra imprimir né" << std::endl;
    return 0;
}
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
```

Portal

```
rgm47006@l4m04u: ~/Documentos/Trabalho_1
rgm47006@l4m04u:~/Documentos/Trabalho_1$ ./servidor
sqrt.cpp #include <iostream>
#include <cmath>

int main(){
    std::cout << sqrt(25) << std::endl;
    return 0;
}
com_erro.cpp int main(){
    std::cout << "Era pra imprimir né" << std::endl;
    return 0;
}
```

Servidor 1

```
rgm47006@l4m21u: ~/Documentos/Trabalho_1
rgm47006@l4m21u:~/Documentos/Trabalho_1$ ./servidor
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
```

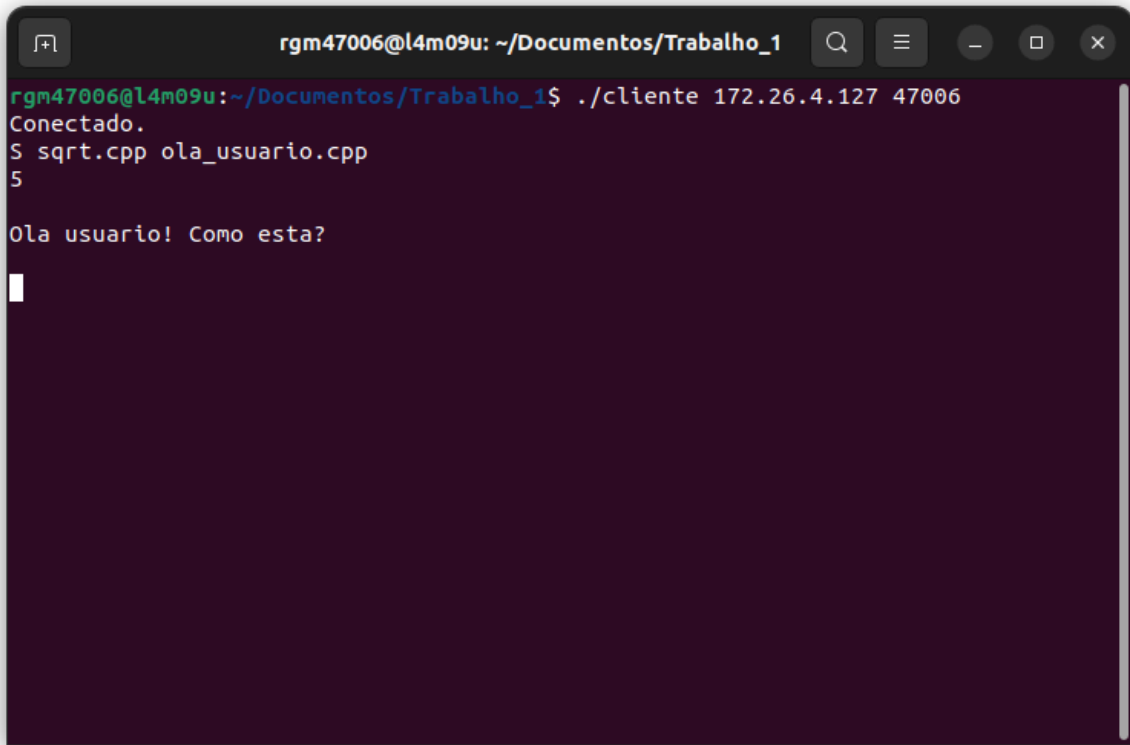
Servidor 2

```
rgm47006@l4m22u: ~/Documentos/Trabalho_1
rgm47006@l4m22u:~/Documentos/Trabalho_1$ ./servidor
com_erro.cpp int main(){
    std::cout << "Era pra imprimir né" << std::endl;
    return 0;
}
```

Servidor 3

6.2 Utilizando Escalonamento Aleatório

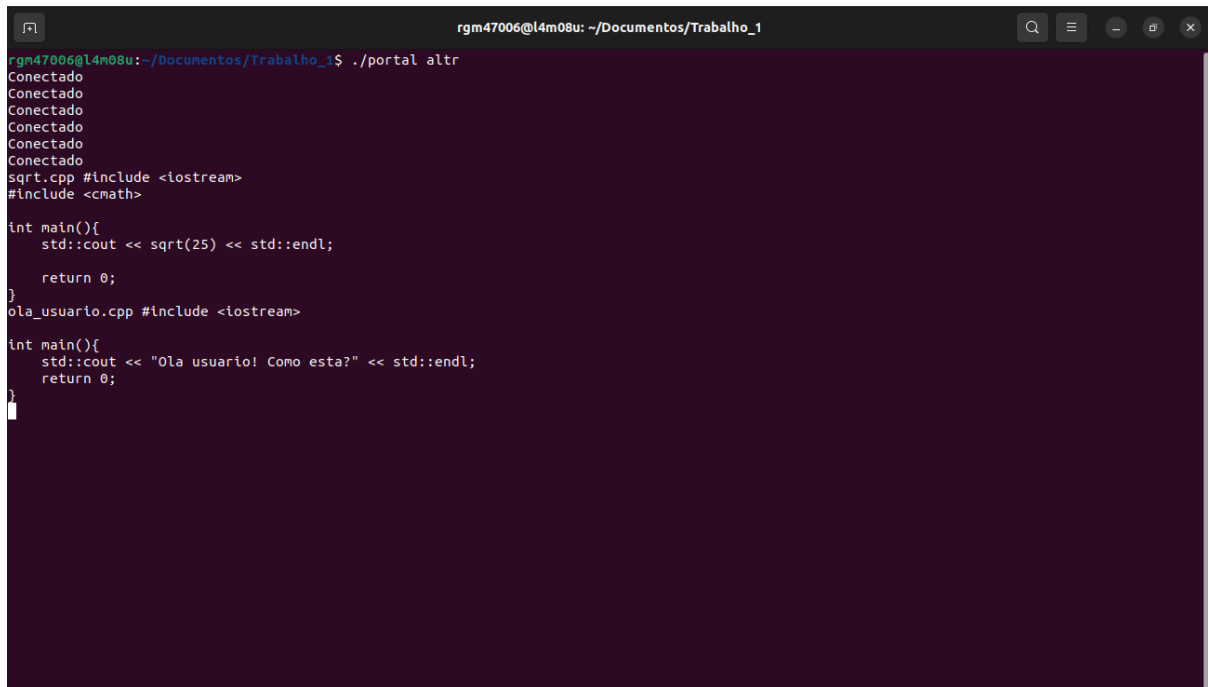
Aqui segue o teste utilizando somente um cliente:



```
rgm47006@l4m09u: ~/Documentos/Trabalho_1
rgm47006@l4m09u:~/Documentos/Trabalho_1$ ./cliente 172.26.4.127 47006
Conectado.
S sqrt.cpp ola_usuario.cpp
5

Ola usuario! Como esta?
```

Cliente

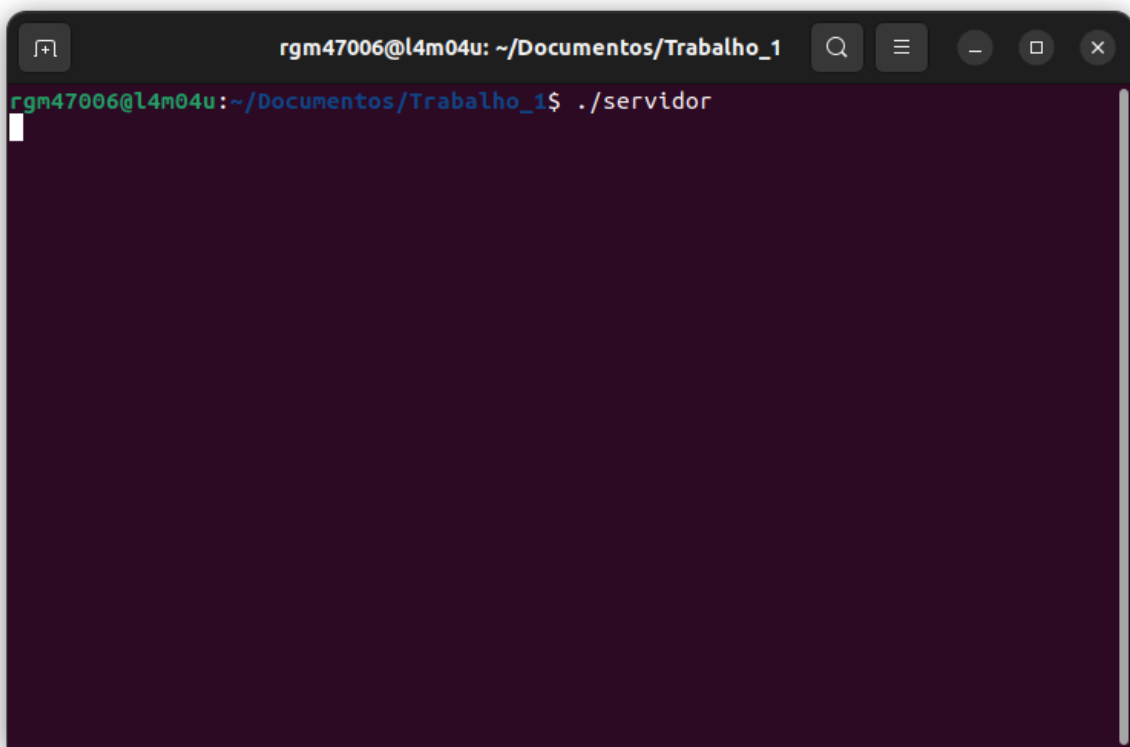


```
rgm47006@l4m08u: ~/Documentos/Trabalho_1
rgm47006@l4m08u:~/Documentos/Trabalho_1$ ./portal altr
Conectado
Conectado
Conectado
Conectado
Conectado
Conectado
Conectado
sqrt.cpp #include <iostream>
#include <cmath>

int main(){
    std::cout << sqrt(25) << std::endl;
    return 0;
}
ola_usuario.cpp #include <iostream>

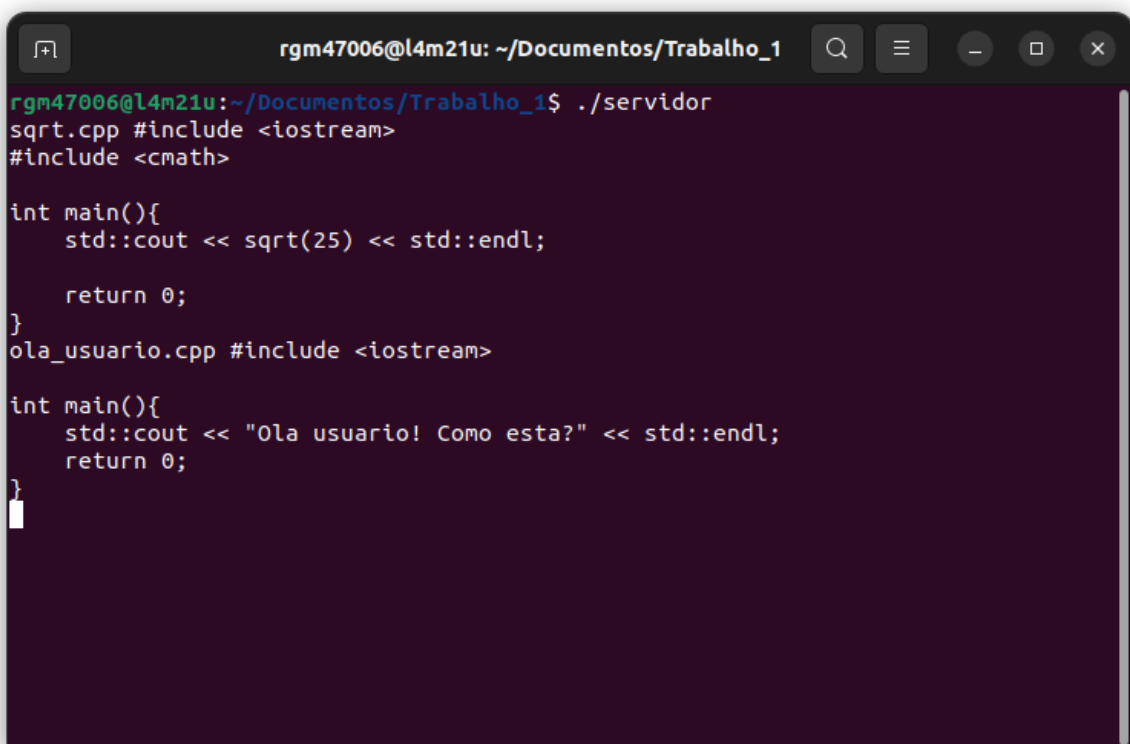
int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
```

Portal

A terminal window with a dark purple background. The title bar shows the user 'rgm47006@l4m04u' and the directory '~/Documentos/Trabalho_1'. The command prompt shows the user has executed './servidor' and the cursor is on the next line.

```
rgm47006@l4m04u: ~/Documentos/Trabalho_1
rgm47006@l4m04u:~/Documentos/Trabalho_1$ ./servidor
```

Servidor 1

A terminal window with a dark purple background. The title bar shows the user 'rgm47006@l4m21u' and the directory '~/Documentos/Trabalho_1'. The command prompt shows the user has executed './servidor'. Below this, two C++ code snippets are displayed. The first snippet is for 'sqrt.cpp' and the second is for 'ola_usuario.cpp'.

```
rgm47006@l4m21u: ~/Documentos/Trabalho_1
rgm47006@l4m21u:~/Documentos/Trabalho_1$ ./servidor
sqrt.cpp #include <iostream>
#include <cmath>

int main(){
    std::cout << sqrt(25) << std::endl;

    return 0;
}
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
```

Servidor 2


```
rgm47006@l4m22u: ~/Documentos/Trabalho_1
rgm47006@l4m22u:~/Documentos/Trabalho_1$ ./servidor
```

Servidor 3

E agora, o caso de teste com dois clientes simultâneos:

```
rgm47006@l4m09u: ~/Documentos/Trabalho_1
rgm47006@l4m09u:~/Documentos/Trabalho_1$ ./cliente 172.26.4.127 47006
Conectado.
$ sqrt.cpp com_erro.cpp ola_usuario.cpp
5

com_erro.cpp: In function 'int main()':
com_erro.cpp:2:10: error: 'cout' is not a member of 'std'
  2 |         std::cout << "Era pra imprimir né" << std::endl;
    |         ^~~~~
com_erro.cpp:1:1: note: 'std::cout' is defined in header '<iostream>'; did you forget to '#include <iostream>'?
+++ |+#include <iostream>
  1 | int main(){
com_erro.cpp:2:48: error: 'endl' is not a member of 'std'
  2 |         std::cout << "Era pra imprimir né" << std::endl;
    |                                                ^~~~~
com_erro.cpp:1:1: note: 'std::endl' is defined in header '<ostream>'; did you forget to '#include <ostream>'?
+++ |+#include <ostream>
  1 | int main(){

Ola usuario! Como esta?
```

Cliente 1

```
rgm47006@l4m10u: ~/Documentos/Trabalho_1
rgm47006@l4m10u:~/Documentos/Trabalho_1$ ./cliente 172.26.4.127 47006]
Conectado.
S com_erro.cpp ola_usuario.cpp
com_erro.cpp: In function 'int main()':
com_erro.cpp:2:10: error: 'cout' is not a member of 'std'
  2 |         std::cout << "Era pra imprimir né" << std::endl;
    |         ^~~~~
com_erro.cpp:1:1: note: 'std::cout' is defined in header '<iostream>'; did you forget to '#include <iostream>'?
+++ |+#include <iostream>
  1 | int main(){
com_erro.cpp:2:48: error: 'endl' is not a member of 'std'
  2 |         std::cout << "Era pra imprimir né" << std::endl;
    |                                                ^~~~~
com_erro.cpp:1:1: note: 'std::endl' is defined in header '<ostream>'; did you forget to '#include <ostream>'?
+++ |+#include <ostream>
  1 | int main(){

Ola usuario! Como esta?
█
```

Cliente 2

```
rgm47006@l4m08u: ~/Documentos/Trabalho_1
rgm47006@l4m08u:~/Documentos/Trabalho_1$ ./portal altr
Conectado
Conectado
Conectado
Conectado
Conectado
Conectado
Conectado
Servidor escolhido: 0
com_erro.cpp int main(){
    std::cout << "Era pra imprimir né" << std::endl;
    return 0;
}
Servidor escolhido: 2
sqrt.cpp #include <iostream>
#include <cmath>

int main(){
    std::cout << sqrt(25) << std::endl;

    return 0;
}
Servidor escolhido: 1
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
Servidor escolhido: 0
com_erro.cpp int main(){
    std::cout << "Era pra imprimir né" << std::endl;
    return 0;
}
Servidor escolhido: 1
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
```

Portal

```
rgm47006@l4m04u: ~/Documentos/Trabalho_1
rgm47006@l4m04u:~/Documentos/Trabalho_1$ ./servidor
com_erro.cpp int main(){
    std::cout << "Era pra imprimir né" << std::endl;
    return 0;
}
com_erro.cpp int main(){
    std::cout << "Era pra imprimir né" << std::endl;
    return 0;
}

```

Servidor 1

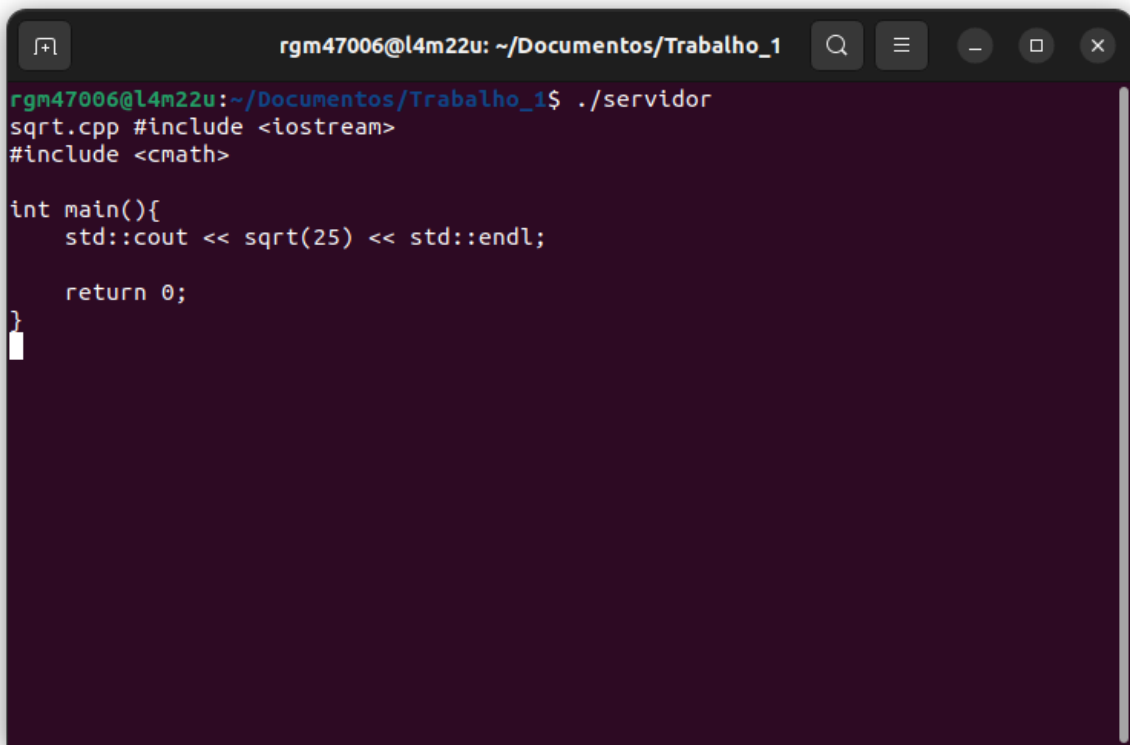
```
rgm47006@l4m21u: ~/Documentos/Trabalho_1
rgm47006@l4m21u:~/Documentos/Trabalho_1$ ./servidor
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}
ola_usuario.cpp #include <iostream>

int main(){
    std::cout << "Ola usuario! Como esta?" << std::endl;
    return 0;
}

```

Servidor 2



A terminal window with a dark background and light-colored text. The window title bar shows the user 'rgm47006@l4m22u' and the directory '~/Documentos/Trabalho_1'. The terminal contains C++ code for a program named 'servidor'. The code includes headers for `<iostream>` and `<cmath>`, and a `main` function that prints the square root of 25 using `std::cout` and `std::endl`, then returns 0.

```
rgm47006@l4m22u: ~/Documentos/Trabalho_1
rgm47006@l4m22u:~/Documentos/Trabalho_1$ ./servidor
sqrt.cpp #include <iostream>
#include <cmath>

int main(){
    std::cout << sqrt(25) << std::endl;

    return 0;
}
```

Servidor 3

7 TESTES CONFORME ESPECIFICAÇÃO

Aqui estão algumas imagens de teste do sistema conforme as especificações do PDF, sem impressão no portal e nos servidores.



```
rgm47006@l4m14u: ~/Downloads/Trabalho_1
./cliente 172.26.4.238 47006
[com_erro.cpp]
[sqrt.cpp]
[ola_usuario.cpp]
$ [sqrt.cpp]
$

$ [sqrt.cpp, ola_usuario.cpp]
$

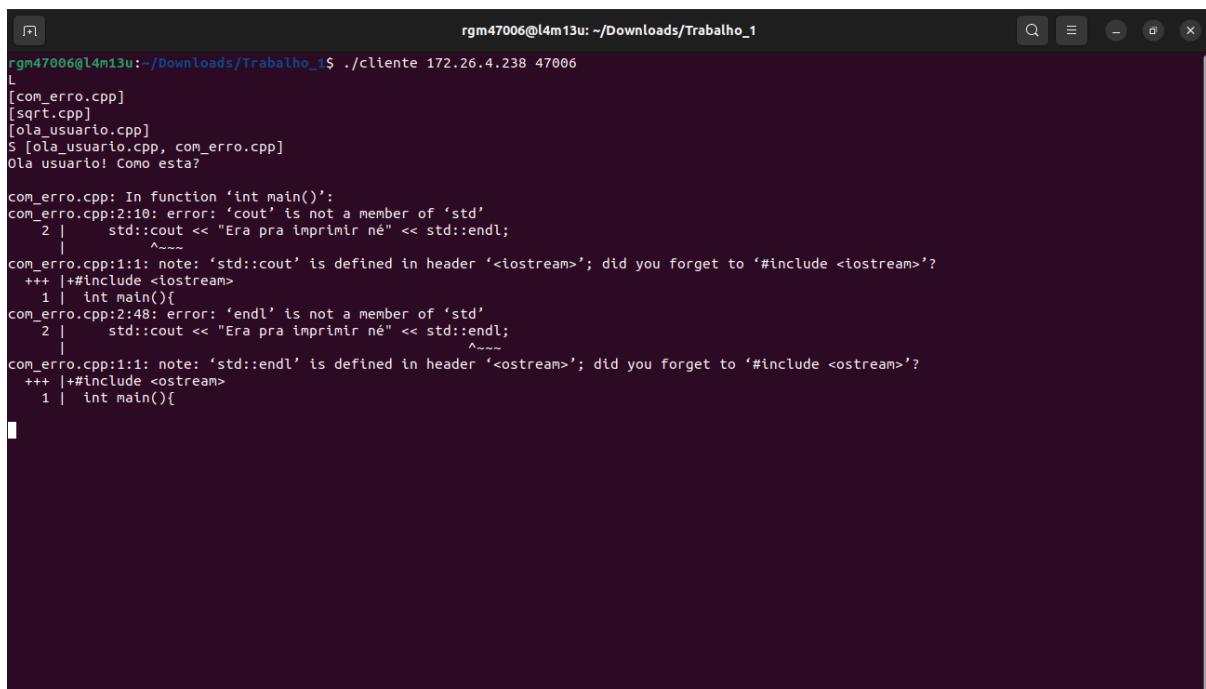
ola usuario! Como esta?

$ [sqrt.cpp, ola_usuario.cpp, com_erro.cpp]
$

ola usuario! Como esta?

com_erro.cpp: In function 'int main()':
com_erro.cpp:2:10: error: 'cout' is not a member of 'std'
    2 |     std::cout << "Era pra imprimir né" << std::endl;
      |           ^~~~~
com_erro.cpp:1:1: note: 'std::cout' is defined in header '<iostream>'; did you forget to '#include <iostream>'?
+++ |+#include <iostream>
    1 | int main(){
com_erro.cpp:2:48: error: 'endl' is not a member of 'std'
    2 |     std::cout << "Era pra imprimir né" << std::endl;
      |                                           ^~~~~
com_erro.cpp:1:1: note: 'std::endl' is defined in header '<ostream>'; did you forget to '#include <ostream>'?
+++ |+#include <ostream>
    1 | int main(){
```

Cliente 1



```
rgm47006@l4m13u: ~/Downloads/Trabalho_1
./cliente 172.26.4.238 47006
[com_erro.cpp]
[sqrt.cpp]
[ola_usuario.cpp]
$ [ola_usuario.cpp, com_erro.cpp]
ola usuario! Como esta?

com_erro.cpp: In function 'int main()':
com_erro.cpp:2:10: error: 'cout' is not a member of 'std'
    2 |     std::cout << "Era pra imprimir né" << std::endl;
      |           ^~~~~
com_erro.cpp:1:1: note: 'std::cout' is defined in header '<iostream>'; did you forget to '#include <iostream>'?
+++ |+#include <iostream>
    1 | int main(){
com_erro.cpp:2:48: error: 'endl' is not a member of 'std'
    2 |     std::cout << "Era pra imprimir né" << std::endl;
      |                                           ^~~~~
com_erro.cpp:1:1: note: 'std::endl' is defined in header '<ostream>'; did you forget to '#include <ostream>'?
+++ |+#include <ostream>
    1 | int main(){
```

Cliente 2

8 CONCLUSÃO

O desenvolvimento deste trabalho foi prazeroso, realmente é algo diferente e mais entusiasmante do que é visto em geral no curso, acredito que tenha me ensinado muito, não só sobre *sockets* e a arquitetura cliente-servidor como também sobre *makefiles* e a linguagem C++ com suas bibliotecas, espero poder implementar também posteriormente escalonamento paralelo, utilizando este sistema já criado.

REFERÊNCIAS

BINARYTIDES. **Get ip address from hostname in C with Linux sockets**. Disponível em: https://www.binarytides.com/hostname-to-ip-address-c-sockets-linux/#google_vignette.

Acesso em: 9 de jul. de 2024.

CURSO MAKEFILE. Disponível em:

<https://www.youtube.com/watch?v=Tt3BLZCVjAE&list=PLLCFxfe9wkl-tCZvSCbzQGcNv9nSN5ZAP>. Acesso em: 9 de jul. de 2024.

GEEKSFORGEEKS. **How to Split a String by Multiple Delimiters in C++?**. Disponível em: <https://www.geeksforgeeks.org/how-to-split-string-by-multiple-delimiters-in-cpp/>.

Acesso em: 9 de jul. de 2024.

GNU MAKE. Disponível em: <https://www.gnu.org/software/make/manual/make.html>.

Acesso em: 9 de jul. de 2024.