

UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

TRABALHO PRÁTICO 3
CLOUD TEXT EM TCP IPV4/IPV6

NOME: MATHEUS VICTOR RAMOS DOS ANJOS

PROFESSOR: LUIZ FILIPE MENEZES VIEIRA

MONITOR: GABRIEL DE BIASI

TURMA: TW2

DATA: 10 DE JULHO

BELO HORIZONTE

2017

INTRODUÇÃO

O trabalho consiste em desenvolver uma aplicação que fornece um serviço de cloud text para clientes utilizando um terminal.

Os programas irão funcionar utilizando o TCP e devem funcionar com IPv4 e IPv6, com o servidor sendo multicliente.

PROBLEMA A SER TRATADO

O problema a ser tratado é a implementação de um servidor que recebesse quatro tipos de comandos e respondesse-os de maneira exata. A seguir, foi copiado da descrição do trabalho os comandos e as respostas para cada caso.

A mensagem de registro (N) deverá ser lida da seguinte maneira:

N [nome_do_usuario] [senha]

Exemplo: **N gabriel 12345**

Ao receber uma mensagem (N), o servidor precisará verificar se já não existe um usuário com esse identificador. O servidor deverá responder o cliente informando o sucesso ou não da operação.

O programa do cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de registro:

Mensagem na Tela	Explicação
N 0	Sucesso, usuário criado.
N -1	Erro, usuário já existe.

A mensagem de envio (S) deverá ser lida da seguinte maneira:

S [nome_do_usuario] [senha] [nome_do_arquivo] [conteudo_do_arquivo]

Exemplo: **S gabriel 12345 meu_diario** Esta será uma mensagem importante

Ao receber uma mensagem (S), o servidor precisará verificar se já existe uma mensagem com este mesmo nome (e mesmo assim substituir seu conteúdo), verificar se existe o usuário e verificar a senha.

O programa do cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de envio:

Mensagem na Tela	Explicação
S 1	Sucesso, mensagem foi salva porém substituiu um conteúdo já existente.
S 0	Sucesso, mensagem salva.
S -1	Erro, usuário não existe.
S -2	Erro, senha incorreta.

A mensagem de recebimento (R) deverá ser lida da seguinte maneira:

R [nome_do_usuario] [senha] [nome_do_arquivo]

Exemplo: **R gabriel 12345 meu_diario**

Ao receber uma mensagem (R), o servidor precisará verificar se a mensagem existe, se o usuário existe e se a senha confere e enviar para o cliente o resultado do comando e o conteúdo da mensagem.

O programa do cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de recebimento:

Mensagem na Tela	Explicação
R 0 [mensagem]	Sucesso, no lugar de [mensagem], coloque o conteúdo da mensagem.
R -1	Erro, usuário não existe.
R -2	Erro, senha incorreta.
R -3	Erro, mensagem não encontrada.

A mensagem de lista (L) deverá ser lida da seguinte maneira:

L [nome_do_usuário] [senha]

Exemplo: **R gabriel 12345**

Ao receber uma mensagem (L), o servidor precisará verificar se o usuário existe e se a senha confere.

O programa do cliente deverá imprimir na tela **APENAS** as seguintes mensagens após a tentativa do comando de lista:

Mensagem na Tela	Explicação
L 0 [arquivo1] [arquivo 2] [...]	Sucesso, no lugar de [arquivo1], coloque o nome dos arquivos. Se não houver nenhum, deixe apenas "L 0".
L -1	Erro, usuário não existe.
L -2	Erro, senha incorreta.

ALGORITMOS, FUNÇÕES, PROCEDIMENTOS E DECISÕES DE IMPLEMENTAÇÃO

• ALGORITMO

Para a implementação dos códigos, foram utilizados um main para o cliente, um main para o servidor e uma biblioteca "files.h" que o servidor utilizou para tratar as mensagens.

○ Servidor

O main servidor consiste em uma configuração de um server TCP multiclente utilizando threads para cada conexão, onde cada cliente recebia um processo diferente. Os serviços que o servidor oferece para seus clientes foram implementados na biblioteca "files.h" que recebia a mensagem do cliente, realizava as operações, e retornava uma mensagem de resposta, de modo que o servidor só recebe a mensagem, analisa qual comando é dado e envia para uma função específica da biblioteca. Para mensagens que não eram de nenhum comando, o servidor responde com uma mensagem vazia e na tela do cliente, apenas aparece um salto de linha, indicando que não foi efetuada nenhuma operação e o servidor não entendeu a mensagem.

○ Cliente

O main do cliente é simples e não possui nenhuma informação a respeito do servidor ou da biblioteca. Ele apenas configura um cliente TCP que aceita tanto IPv4 quanto IPv6. Em seu loop, ele apenas lê uma mensagem, envia para o servidor, recebe a mensagem e testa se é uma mensagem de finalização, retornando para o começo do laço.

○ Biblioteca Files

A biblioteca files foi implementada utilizando dois arquivos, files.c e files.h. No arquivo files.h, são declaradas as funções da biblioteca e cada função tem um cabeçalho com sua descrição. No arquivo files.c são implementadas as funções declaradas. Todas as funções trabalham utilizando arquivos, visto que foi a melhor

opção para lidar com textos, já que em C a utilização de strings é bem complicada, o que torna inviável a utilização de um tipo abstrato de dados. A organização é dividida em um arquivo com o registro de usuários e senhas, um arquivo para cada cliente, e para cada texto de cada cliente, um arquivo diferente.

Cada cliente possui um arquivo onde estão listados seus arquivos de texto. Como cada cliente possui um único nome, cada arquivo de texto é concatenado com o nome de seu dono, para que vários usuários possam utilizar mesmos nomes para seus arquivos sem conflito.

Funções da biblioteca files.h

1. Nome: New_user

Entrada: char* login, char* senha

Saída: char* result

Descrição: Recebe um usuário e senha e cadastra eles no arquivo users.txt ou retorna -1 caso o usuário já exista

2. Nome: Sendmsgtp

Entrada: char *login, char *senha, char *book, char *message

Saída: char *buffer

Descrição: Envia uma mensagem de upload de conteúdo de texto ou sinaliza erro.

3. Nome: Begintp

Entrada: void

Saída: void

Descrição: Cria arquivo inicial de usuários e senhas.

4. Nome: Readmsgtp

Entrada: char *login, char *senha, char *book

Saída: char* result

Descrição: Recebe usuário, senha, nome do arquivo e exibe o conteúdo do arquivo ou em caso de erro, sinaliza-o.

5. Nome: List

Entrada: char *login, char *senha

Saída: char *result

Descrição: Recebe login e senha e retorna os arquivos de texto do usuário ou então reporta erro.

TRATAMENTO IPv4 E IPv6

O servidor deve funcionar tanto para clientes com IPv4 quanto para IPv6. Para isso, no código "cliente.c" foi usada a estrutura genérica addrinfo e a família do endereço foi deixada como não especificada para aceitar qualquer tipo.

Podendo assim aceitar qualquer família de endereço. No servidor, foi utilizada a estrutura sockaddr_in6 e o socket foi configurado para receber tanto conexões com IPv4 e IPv6.

No servidor, foram implementadas threads para tratar vários clientes em paralelo. Para isso, foi utilizada a biblioteca pthread.h e no Makefile, foi adicionado o comando -pthread na hora de compilar o servidor.

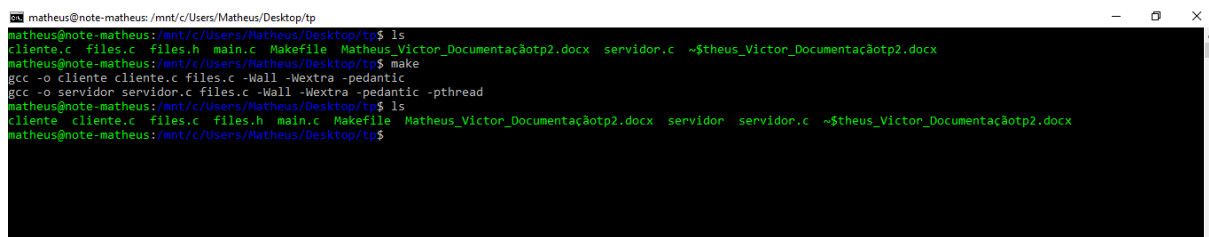
Para cada cliente, é criado um socket novo, que não precisa mais se comunicar com o segmento principal. Após isso, dentro do novo processo, o socket do servidor principal é liberado e a conexão com o cliente começa com o próprio socket. Quando o cliente encerra a conexão, o socket é liberado o processo encerrado, enquanto o servidor principal fica sempre esperando novas conexões e transferindo-as para novos processos.

TESTES

Para testar, foi utilizado o Bash Ubuntu on Windows para evitar a instalação de bibliotecas não nativas no gcc do Windows que são necessárias para a comunicação.

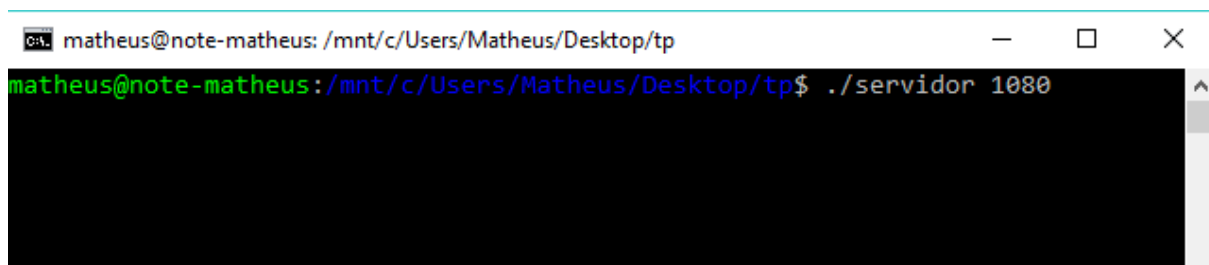
Foi testado utilizando o localhost com IPv6: ":::1" e a porta 1080.

Compilando os códigos utilizando o comando make:



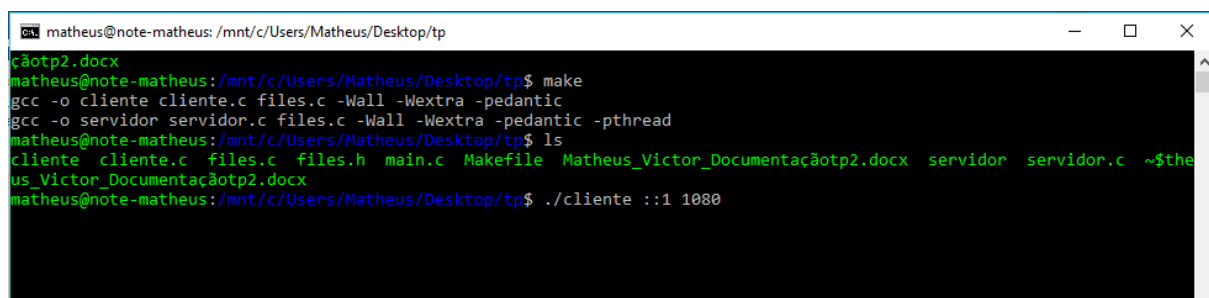
```
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ls
cliente.c files.c files.h main.c Makefile Matheus_Victor_Documentaçãootp2.docx servidor.c ~$theus_Victor_Documentaçãootp2.docx
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ make
gcc -o cliente cliente.c files.c -Wall -Wextra -pedantic
gcc -o servidor servidor.c files.c -Wall -Wextra -pedantic -pthread
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ls
cliente cliente.c files.c files.h main.c Makefile Matheus_Victor_Documentaçãootp2.docx servidor servidor.c ~$theus_Victor_Documentaçãootp2.docx
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$
```

Inicializando o Server



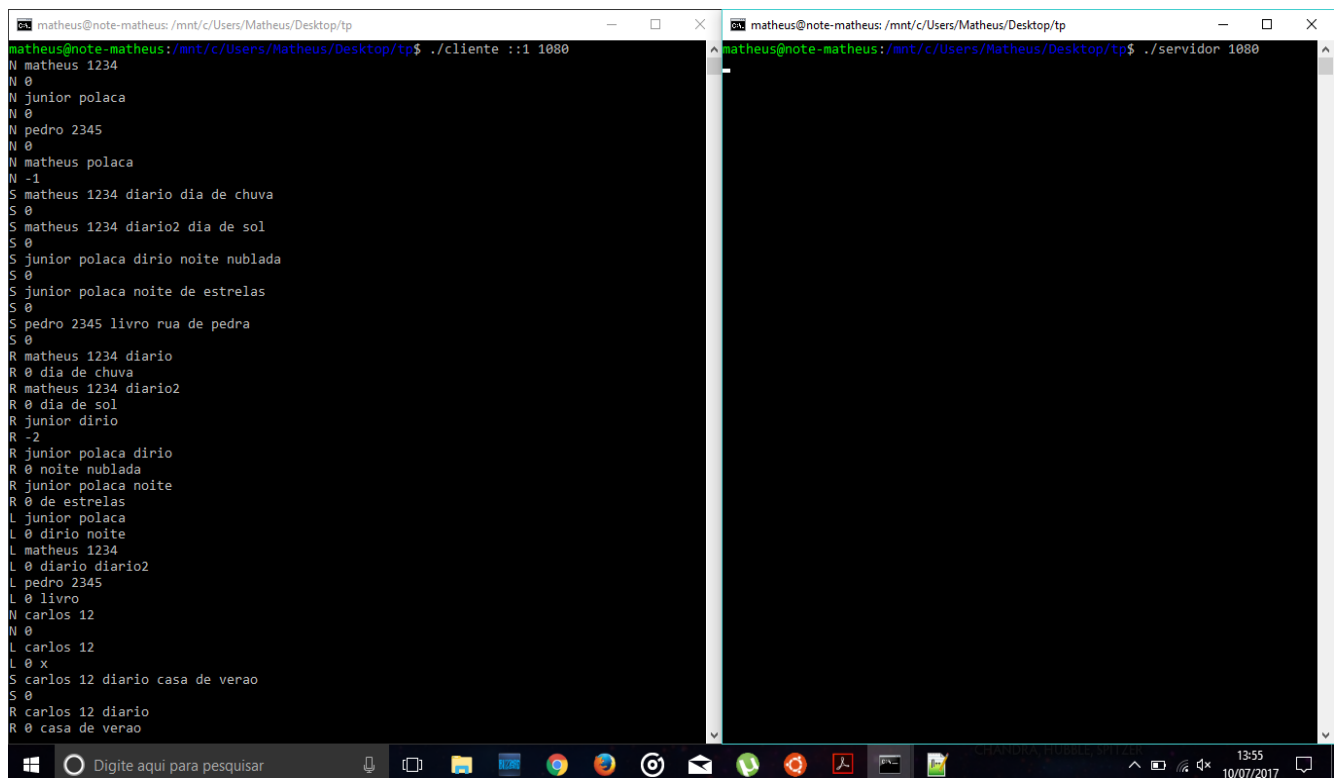
```
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ./servidor 1080
```

Inicializando o Cliente



```
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ make
gcc -o cliente cliente.c files.c -Wall -Wextra -pedantic
gcc -o servidor servidor.c files.c -Wall -Wextra -pedantic -pthread
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ls
cliente cliente.c files.c files.h main.c Makefile Matheus_Victor_Documentaçãootp2.docx servidor servidor.c ~$theus_Victor_Documentaçãootp2.docx
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ./cliente :::1 1080
```

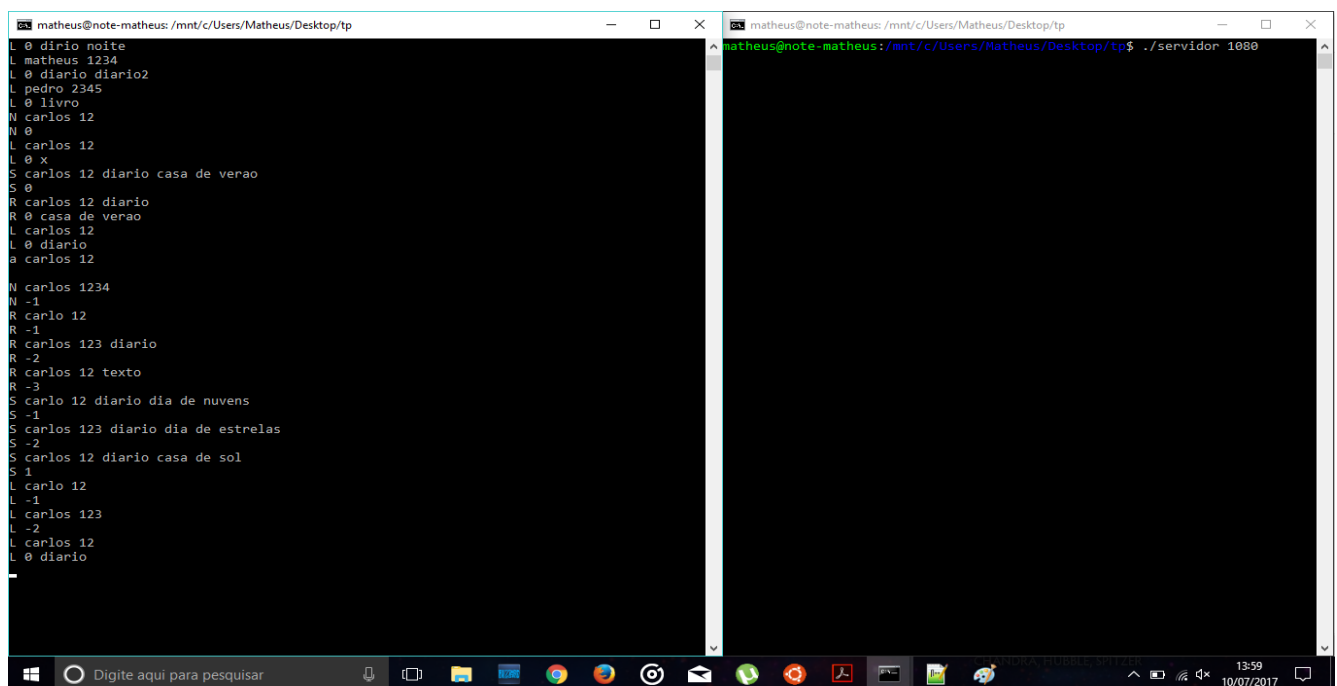
Enviando uma mensagem de cadastro (N), envio (S), leitura (R) e lista (L)



```
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ./cliente :1 1080
N matheus 1234
N 0
N junior polaca
N 0
N pedro 2345
N 0
N matheus polaca
N -1
S matheus 1234 diario dia de chuva
S 0
S matheus 1234 diario2 dia de sol
S 0
S junior polaca dirio noite nublada
S 0
S junior polaca noite de estrelas
S 0
S pedro 2345 livro rua de pedra
S 0
R matheus 1234 diario
R 0 dia de chuva
R matheus 1234 diario2
R 0 dia de sol
R junior dirio
R -2
R junior polaca dirio
R 0 noite nublada
R junior polaca noite
R 0 de estrelas
L junior polaca
L 0 dirio noite
L matheus 1234
L 0 diario diario2
L pedro 2345
L 0 livro
N carlos 12
N 0
L carlos 12
L 0 x
S carlos 12 diario casa de verao
S 0
R carlos 12 diario
R 0 casa de verao
L carlos 12
L 0 diario
a carlos 12

matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ./servidor 1080
```

Enviando mensagens para testar os erros



```
matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ./cliente :1 1080
L 0 dirio noite
L matheus 1234
L 0 diario diario2
L pedro 2345
L 0 livro
N carlos 12
N 0
L carlos 12
L 0 x
S carlos 12 diario casa de verao
S 0
R carlos 12 diario
R 0 casa de verao
L carlos 12
L 0 diario
a carlos 12

matheus@note-matheus: /mnt/c/Users/Matheus/Desktop/tp$ ./servidor 1080
```

Durante os testes de erros, tem uma linha em branco, que é a resposta do servidor à uma mensagem inválida que não comece com a letra correta. O servidor é case sensitive, ou seja, diferencia letras minúsculas de maiúsculas, logo, não aceita as minúsculas.

O servidor não imprime nada na tela a não ser que aconteça algum erro na hora de conectar, como por exemplo, um erro de argumento ou um socket error.

CONCLUSÃO

O trabalho prático foi útil para visualizar a comunicação TCP entre cliente e servidor utilizando dois tipos diferentes de endereçamento IP, além de desenvolver uma aplicação para ser utilizada em rede, que exigiu uma implementação muito bem organizada das funções e procedimentos.

O servidor sempre responde corretamente às mensagens que são enviadas corretamente, embora seja imprevisível como ele responderá com mensagens errôneas, pois o programa poderá ler lixo e utilizar esse conteúdo para operar.

Foi possível testar vários clientes em paralelo, embora no teste apareça apenas um, pois ficou difícil visualizar a os comandos com mais de duas janelas abertas.

O trabalho foi satisfatório e cumpriu o que foi solicitado pela descrição.

REFERÊNCIAS BIBLIOGRÁFICAS

Fclose

<http://manpages.ubuntu.com/manpages/precise/pt/man3/fclose.3.html>

Acesso em: 10 de Julho de 2017

Undefined reference to pthread create in linux

<https://stackoverflow.com/questions/1662909/undefined-reference-to-pthread-create-in-linux>

Acesso em: 10 de Julho de 2017

A Biblioteca String.h

<http://linguagemc.com.br/a-biblioteca-string-h/>

Acesso em: 09 de Julho de 2017

Escrevendo em Arquivos – As funções fputc, fprintf e fputs

<http://www.cprogressivo.net/2013/11/Escrevendo-em-arquivos-As-funcoes-fputc-fprintf-fputs.html>

Acesso em: 09 de Julho de 2017

Operação com arquivos – FILE, fopen, modos de abertura, modos de fechamento, modo texto, binário, EOF, fclose, fcloseall

<http://www.cprogressivo.net/2013/10/Operacoes-com-arquivos-FILE-fopen-modos-de-abertura-fechamento-modo-texto-binario-EOF-fclose-fcloseall.html>

Acesso em: 09 de Julho de 2017

Como ler em arquivos em C – As funções fgetc, fscanf, fgets

<http://www.cprogressivo.net/2013/11/Como-ler-arquivos-em-C-As-funcoes-fgetc-fscanf-fgets.html>

Acesso em: 09 de Julho de 2017