2019

Sistemas Operativos 2018/19

TRABALHO PRÁTICO - META 3

Daniel Manta – 21240652 P4
Pedro Rei – 21240654 P4

Índice

Estruturas de dados

Cliente-defaults.h

```
#define MAX_NOME 9



typedef struct cliente{
        char username[MAX_NOME];
        int pid;
        int valido;
        int slot;
}cli;
```

- Estrutura de dados onde é guardada toda a informação de um cliente para que seja efectuado o seu login sendo o username o nome do cliente, pid o pid do cliente , valido e slot que são duas flags direcionadas à validação do cliente.

Comandos.h

```
void settings(int lin,int col,char *filename,int maxuser, char *mainpipe, int nrpipes);
```
- função pertencente ao comando settings

```
void load(char *filename);
```
-função pertencente ao comando load

```
void save(char *filename);
```
-função pertencente ao comando save

```
void users(serv *ativo, int total);
```
-função pertencente ao comando users

```
void text();
```
-função pertencente ao comando text

Medit-defaults.h

```
#define LIN 15   - Variavel por omissão com o valor máximo de linhas para a matriz do editor.
#define COL 45  - Variavel por omissão com o valor máximo de colunas para a matriz do editor.

typedef struct medit_data{
        char texto[LIN][COL];           -estrutura da matriz medit
        int linhas, colunas;
        char nome[LIN][9];              -nomes dos utilizadores a editar linhas
}medit;
```

Servidor-defaults.h

```
#define TAM 9  //tamanho máximo de username
#define MAX_USERS 3
#define MAX_LIMITE_USERS 15
#define DATABASE "meditdb.txt"
#define PIPEPR "pipr"
#define FIFO_CLI "sss%d"
#define NR_PIPES 1

typedef struct server_users{
        char username[TAM];
        char pipe[20];          //nome do pipe do cliente
        char comunica[10];      //nome do pipe utilizado para escrever para o servidor
        char linha[45];         //buffer para fazer alterações
        char buffer[45];        //buffer para repor no caso de ESC
        int indice;             //indice no vector de ocupação dos pipes
        int pid;
        int servpid;
        int posx;
        int posy;
        int hora, min, seg;     //guardar tempo de chegada do utilizador
        int linhas_editadas;
        int tecla;
        int erro;                       //erro ortográfico
        int emLinha;    //para saber se linha está trancada ou não

}serv;


typedef struct thread_info{
        char pipes[MAX_LIMITE_USERS][3];    //nomes de named pipes usados pelas
threads
        int ocupados[MAX_LIMITE_USERS];     //informação sobre quantos clientes estão a
usar os pipes
        int sair;                           //flag de saida dos pipes
}tr_info;
```

- Estrutura de dados onde é guardada a informação dos utilizadores que se encontrem activos no medit para editar a matriz, onde há o username destes, o named pipe do cliente, o pid do cliente, as posições do cliente e a tecla que este terá carregado. No futuro deverá haver também o momento em que cada terá logado tal como uma ligação para outra estrutura de utilizadores.

-Estrutura de informação pertinente ao funcionamento das threads e na criação dos pipes.

Sinais

Cliente:

SIGINT- O objectivo é teminar o programa.
É enviado no caso de haver comportamentos inesperados.
É enviado pelo utilizador ao cliente.
O cliente fecha o pipe que criou, avisa o servidor e termina.

SIGUSR1-Atualizações à estrutura de dados do medit por parte de outros clientes.
É enviado quando são feitas alterações por um dos outros clientes.
É enviado pelo servidor ao cliente.
O cliente recebe a atualização e atualiza o ecran.

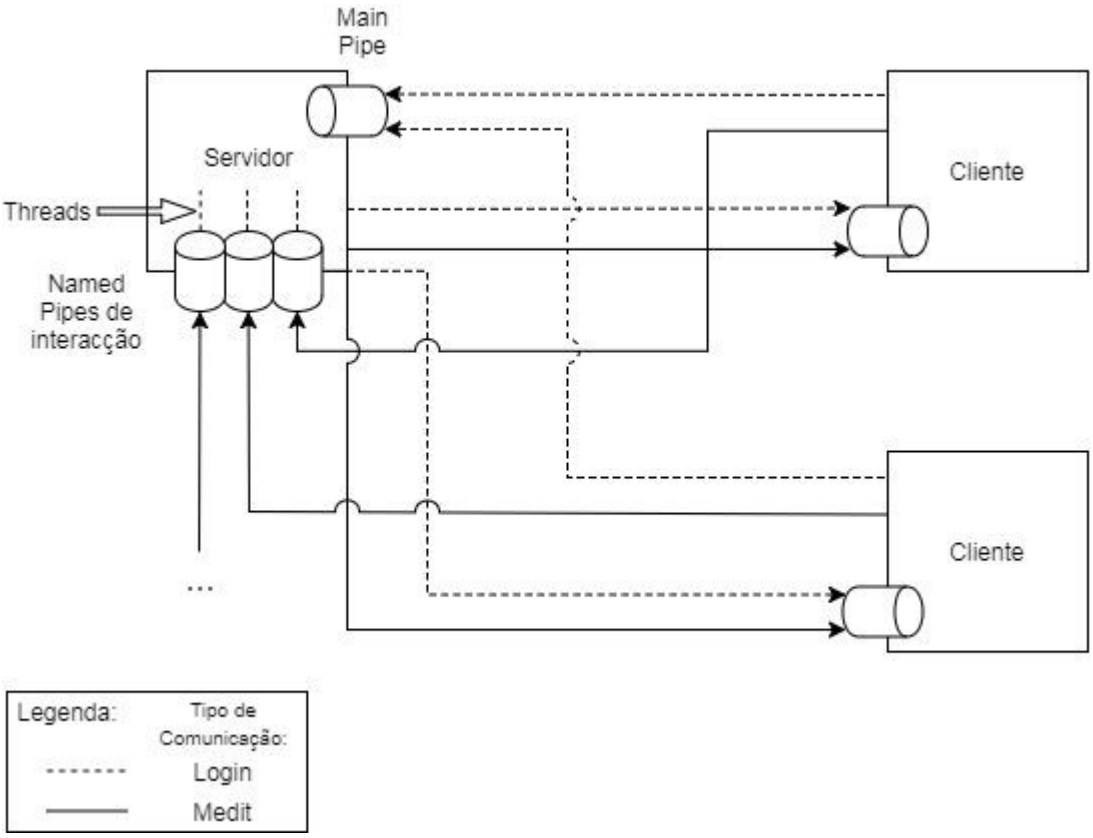
SIGUSR2-O objectivo é terminar o programa.
É recebido quando o servidor termina.
É enviado pelo servidor ao cliente.
O cliente fecha o pipe que criou e termina.

Servidor:

SIGINT-O objectivo é teminar o programa.
É enviado no caso de haver comportamentos inesperados.
É enviado pelo utilizador ao servidor.
O servidor fecha todos os pipes que criou, avisa todos os clientes activos e termina.

SIGUSR1-O objectivo é ter conhecimento da saida de um cliente
É enviado pelo cliente ao servidor.
O servidor atualiza a estrutura dos cliente activos.

Arquitetura Pipes e threads



Este diagrama representa de uma forma simplificada a arquitectura utilizada neste trabalho.

O servidor cria o pipe principal que utiliza para ler as informações úteis para fazer as autenticações(struct cliente) relativas a clientes. De seguida escreve para o cliente o resultado da autenticação que, no caso de sucesso, comunica também as informações relevantes para a utilização do Medit. Isto inclui a designação do named pipe que o cliente passará a usar para escrever ao servidor. Os named pipes utilizados para receber comunicações de clientes relativas ao Medit são todos criados pelo servidor em conjunto com o lançamento de uma thread(linha de execução) para cada um. Em relação às threads, resumidamente, o servidor tem a thread principal que trata da leitura de comandos, uma thread associada à leitura do pipe principal e finalmente x número de threads associadas aos named pipes de interacção.

O cliente por usa vez, cria o named pipe que utilizará para receber as comunicações feitas pelo servidor.

Todos os pipes criados são eliminados por quem os cria.

Os named pipes criados pelo servidor são abertos ao inicio e fechados no fim da vida do servidor. O servidor abre e fecha os pipes dos clientes à medida que vai fazendo as escritas.

O named pipes pelos vários criados pelos clientes são abertos ao inicio e fechado no fim de vida do cliente. Os clientes abrem e fecham os pipes à medida que fazem as escritas.

O servidor cria também dois pipes anónimos, um para leitura e um para escrita, para o aspell. O aspell é lançado através de um fork() e de um exec() que dão origem a processo filho que funciona enquanto servidor se encontrar activo e comunica com o servidor através de redirecionamento.

Existe mais uma thread que é criada e destruida pelo servidor que serve para o funcionamento do comando statistics.

Mecanismos de sincronização

Foram utilizados mutexes.

São necessários para os vários clientes não acederem ao mesmo espaço de memória em simultaneo.

A matriz do medit que apresenta o texto é partilhada por todos os clientes, só deve ser acedida por um cliente de cada vez.

Os mutexes foram utilizados sempre que é feita uma alteração na estrutura de dados do medit.

Funcionalidades realizadas

Os requisitos terão sido maioritareamente realizados com excepção dos comandos save , load e o timeout em linha activa.

O aspell não tem o comportamento 100% desejado, havendo vezes em que a validação ortográfica não funciona correctamente.

Verificação e validação

Terão sido realizados testes à aplicação primáriamente no editor pois terá havido vários bugs que foram resolvidos com sucesso, por exemplo, devido às alterações feitas desde a primeira meta e à adição de pipes terá sido necessário certificar que o servidor faz todos os passos de alteração da matriz, sendo o apenas avisado das alterações que teram sido efectuadas continuamente.

Terão sido realizados enumeros testes à aplicação quer em termos da validação de clientes onde inicialmente apenas um username era aceite tal como na matriz onde esta acedia locais da memória onde não era suposto devido à falta de uma receção(read) por parte do cliente.

Foram feitos testes para confirmar que quando o servidor terminava avisava os clientes devidamente e vice-versa.

Foi testado o funcionamento de multiplos clientes em simultaneo.

Haverão bugs que por falta de experiencia ou de memória que não se encontram aqui descritos.

Comportamentos anómalos conhecidos

Atualmente são conhecidos bugs como quando apagamos um caracter a meio de uma palavra ocasionalmente um caracter que se encontra à frente tambem é apagado e ocasionalmente o aspell aceita palavras que não se encontram corretamente escritas no entanto uma larga maioria das vezes efectua o que lhe é pedido devidamente.

O tempo de sessão dos clientes que é mostrado no comando users mostra temporariamente informação errada, isto é, durante 10 segundos de cada minuto aparece informação incorrecta.

Anexos

Aqui serão colocados todos os ficheiros de código que terão sido utilizados nesta meta:

- Makefile
- Meditdb.txt
- Cliente-defaults.h
- Medit-defaults.h
- Servidor-defaults.h
- Comandos.h
- Comandos.c
- Servidor.c
- Cliente.c

Alterações relativamente a metas anteriores

Houve algumas alterações desde a meta 1 até à meta 2 sendo a principal alteração a eliminação dos ficheiros medit.c e medit.h pois estes passaram a estar integrados no ficheiro cliente.c devido a tudo o que terá sido feito com a biblioteca ncurses.h ser a interface de cada cliente ativo.

Terão sido adicionados ao:

- Cliente-defaults.h variaveis na estrutura, nomeadamente as variáveis valido e slot na estrutura presente.
- Medit-defaults.h a string line que consiste de um buffer para uma linha da matriz na estrutura presente.
- Servidor-defaults.h as variáveis pid,posx,posy e tecla que são todas elas variáveis de suporte à comunicação por pipes tal como a string pipe onde é guardado o pipe do utilizador em questão. Para além disso foram adicionados os nome por defeito dos pipes do servidor e cliente sendo “ccc” do servidor(FIFO_SERV) e “sss%d” dos clientes(FIFO_CLI),e a adição de uma variavel na estrutura serv chamada de erro com o objectivo de determinar quando existe um erro proveniente do aspell (&).

Meta 3:

- O mecanismo select foi trocado por threads.
- Houve alterações nas estruturas de dados do medit e dos clientes activos.
- Foram completados vários comandos que não teriam sido implementados previamente.