Instituto Superior de Engenharia de Coimbra
Licenciatura em Engenharia Informática
Sistemas Operativos

Relatório SO - Meta 2

2018/2019

João Silva 21270383
Joaquim Santos 21270351

Índice

1. Estrutura de Dados

A segunda meta do trabalho consistia em implementar mecanismos de comunicação entre cliente e servidor, mais propriamente mecanismos *First In First Out*. Por enquanto consideramos que só um cliente acede ao servidor. Para que a comunicação fosse possível estruturas de dados tinham de ser criadas. As que decidimos implementar foram as seguintes:

1.1 Estrutura Valida

A estrutura Valida que visa validar o user que pretende estabelecer "conexão" com o servidor, constituída pelo pid do processo user, uma variável de controlo para dar flexibilidade no código (verificações), o nome do user (parâmetro fundamental para a validação) e o named pipe para onde o cliente vai falar se o user foi verificado como cliente.

```
typedef struct Valida {
    pid_t pid_user;
    int ver; // var de controlo
    char nome[9]; // nome user
    char np_name[20]; // np para onde fala
} valida;
```

1.2 Estrutura Request

A estrutura Request que é responsável por armazenar a informação necessária do cliente para o servidor para que ele possa reservar a linha no editor de texto, se esta estiver desocupada. A estrutura tem como variáveis o pid do cliente que está a fazer a requisição da linha, o número da linha e o seu conteúdo.

```
struct Request {
    pid_t pid_cliente;
    int nr_linha, aspell;
    char texto[1000];
};
```

1.3 Estrutura Controlo

A estrutura Controlo como o nome indica tem como por função reservar variáveis controlo que permitem gerir, de uma forma mais eficiente e simples, o funcionalidades disponíveis ao cliente. É também a estrutura que leva a resposta do servidor ao pedido (Request) do cliente. Esta possui a variável "perm" que indica ao cliente se a linha que ele

quer está livre ou não e a variável sair que indica se ao cliente se o servidor continua ativo ou se já encerrou.

```
struct Controlo {
    int perm; // permissao para editar linha
    int sair; // servidor vai fechar?
};
```

1.4 Estrutura Comunica

A estrutura Comunica envolve as estruturas Request e Controlo para facilitar a comunicação.

```
typedef struct Comunica {
    struct Request request;
    struct Controlo controlo;
} comunica;
```

1.5 Variáveis globais

Neste momento existem no código 4 variáveis globais comuns apenas ao servidor. Estas variáveis são “inter_fifo_fd” que serve para guardar o descritor named pipe de interação para o cliente, ”s_fifo_fd” que serve para guardar o descritor do named pipe principal do servidor, ”pos” que guarda o número do named pipe de interação a usar pelo cliente e “SAIR” variável que controla o ciclo da thread tarefa[1].

2. Funcionalidades Realizadas

Funcionalidade	**Realização**
Implementação do editor de texto (com recurso a ncurses) - **Meta 1**	Totalmente Implementada
Comunicação através de named pipes entre cliente e servidor	Totalmente Implementada
Gestão de clientes	Totalmente Implementada
Adaptação do dicionário Aspell	Totalmente Implementada mas com bugs
Login dos clientes	Totalmente Implementada

3. Named Pipes

A comunicação entre servidor e cliente foi feita com base em mecanismos *First In First Out*, e por isso tivemos de ter em atenção alguns pormenores. Especial cuidado onde as funções sistema era posicionadas para que o programa nunca entre num deadlock. Preocupação com a leitura das estruturas, com verificações para ver se eram feitas na sua totalidade ou não.

O servidor quando é inicializado não tem qualquer tipo de função, fica à espera que um user tente estabelecer conexão consigo, pode porém receber indicações por parâmetro na sua inicialização. Quando o cliente inicia o seu processo, o seu named pipe de validação é criado com o nome "vvv" (VAL_FIFO) + mais o seu pid. A estrutura Valida é então preenchida e depois da abertura do named pipe principal do servidor (SERVER_FIFO_P) é enviada ao servidor para que este possa verificar na sua base de dados se o nome do user existe ou não. Se o utilizador for confirmado então o servidor cria N named pipes de interação para onde o cliente pode falar, sendo lhe atribuído o que tem menos clientes. Essa informação é então inserida na estrutura Valida e escrita no named pipe de validação do user depois da abertura do mesmo para escrita por parte do servidor.

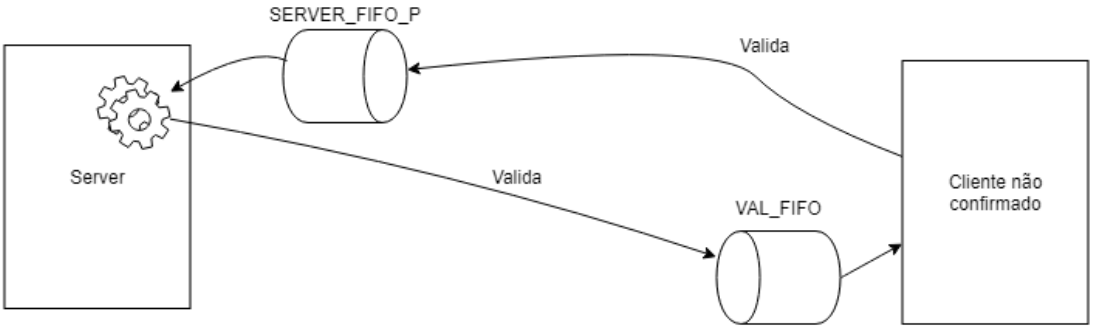

Figura 1 - Interação de server + cliente quando o cliente não existe na base de dados

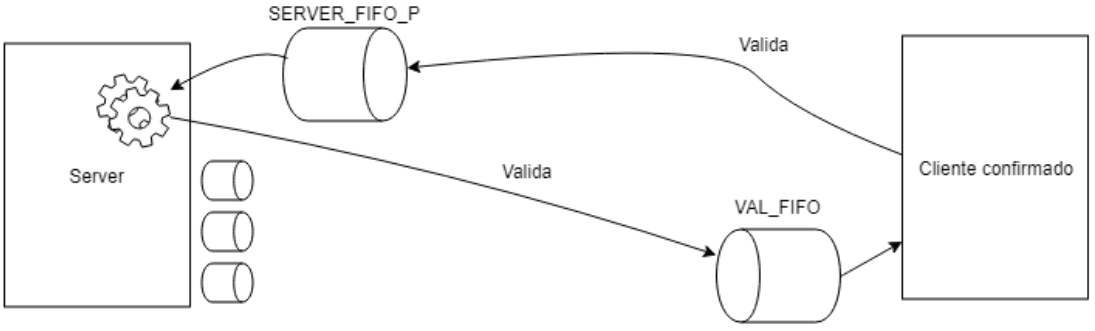

Figura 2 - Interação de server + cliente quando o cliente existe na base de dados

Neste momento o cliente cria o seu named pipe e abre o pipe de interação que recebeu do servidor para escrita.

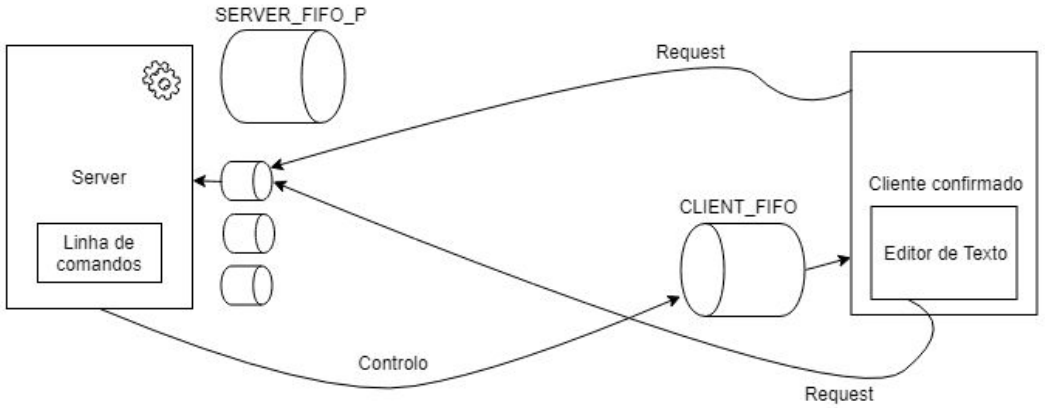

Figura 3 - Interação de servidor - cliente , quando o cliente reserva a sua linha no editor

O acesso ao editor de texto é então concebido e o cliente pode escolher qualquer linha desde que esta esteja livre. Para esse efeito é preenchida uma estrutura Request com a informação necessária (pid do processo cliente, nr_linha, …) e escrita no named pipe de interação atribuído ao cliente anteriormente. O servidor verifica se a linha está vazia e se estiver é atribuída ao cliente. A resposta é mandada ao cliente para o seu named pipe. O cliente passa a poder editar a linha que requisitou, quando acaba e quer sair, desde que não seja de forma forçada ESC, o cliente o cliente irá ser submetido a uma avaliação por parte do aspell, se esta correr bem, ele sairá da edição e as mudanças são feitas, se não correr bem a validação, o cliente é forçado a manter se na linha até que as palavras estejam todas certas, a partir do momento em que saiu da edição, o seu pid será também retirado da tabela de editores.

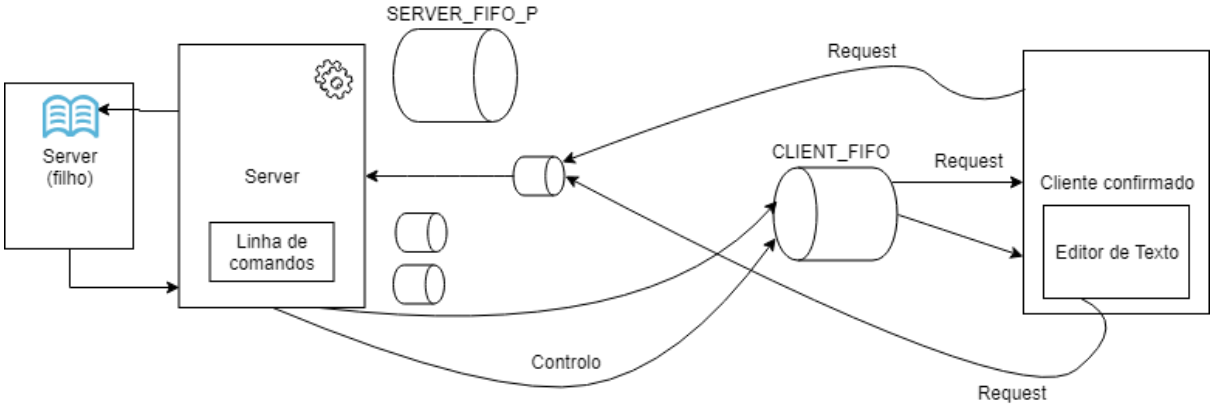

Figura 4 - Comunicação com o servidor para validação do texto editado

4. Verificação e Validação

As verificações nos mecanismos de comunicação baseiam se em imprimir para o ecrã flags a verificar se os named pipes são abertos ou não e os bytes que recebem ou enviam.

5. Comportamentos anómalos conhecidos

Quando um cliente é inicializado e a sua validação não é bem sucedida, se outro cliente tentar entrar no mesmo servidor existem erros.

O programa, ao chamar o aspell para validação do conteúdo nas linhas tem bugs, este comportamento anómalo tem causa na string que é validada, pois esta apresenta lixo no final.

6. Limitações

O servidor ainda não se encontra preparado para receber mais do que um cliente. O conteúdo das linhas não é completamente validado no aspell.