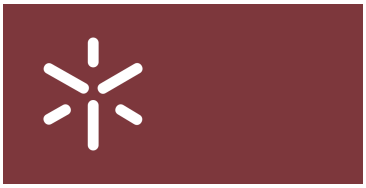

Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Segurança de Sistemas Informáticos

Ano Letivo de 2023/2024

Trabalho Prático 2 - Concordia

Diogo Matos
A100741

Júlio Pinto
A100742

Mário Rodrigues
A100109

May 13, 2024

SSI

Índice

1. Introdução

Este trabalho foi desenvolvido no âmbito da unidade curricular de Segurança de Sistemas Informáticos (SSI). O objetivo deste trabalho foi desenvolver um serviço de email local, denominado `Concordia`, e garantir a sua segurança utilizando permissões de utilizadores do sistema **GNU+LINUX**.

2. Arquitetura

Relativamente à arquitetura do projeto, foi essencial pensar nesta de maneira a conseguirmos obter um sistema seguro. Sendo assim, esta passou por diferentes iterações, porém consideramos que esta foi a melhor solução que conseguimos desenvolver.

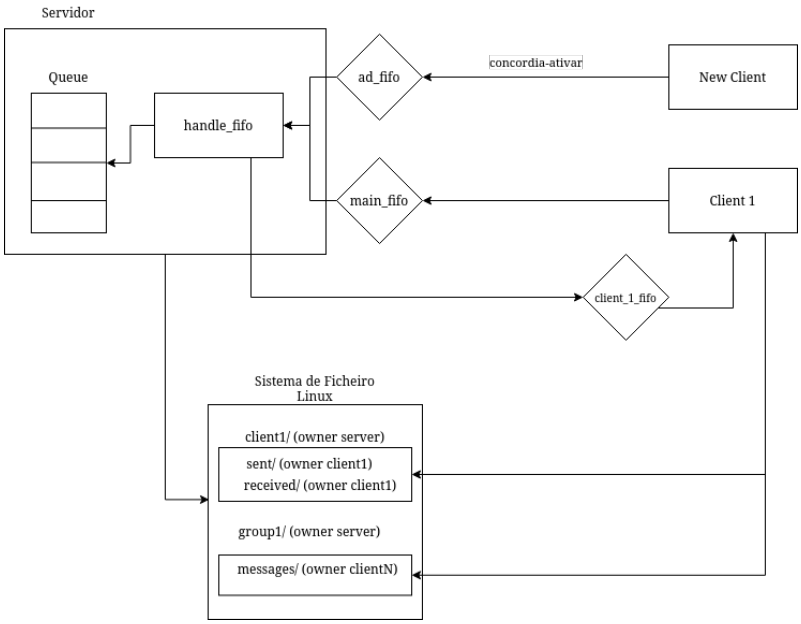

Figura 1: Arquitetura do Sistema

Como podemos ver acima, a nossa arquitetura baseia-se num sistema *cliente-servidor*.

Neste temos que um *New Client* (Cliente por autenticar no sistema), que pretende iniciar-se no serviço, escreve para o nosso `ad_fifo` (*activate/deactivate* FIFO), que é aberto a qualquer utilizador. O servidor, vendo o tipo do pedido, cria então no sistema Linux uma pasta para este, dando-lhe as devidas permissões. O cliente é informado através de um FIFO temporário, criado pelo próprio, que a sua pasta pessoal foi criada e, por sua vez, cria as pastas *sent* e *received*, sendo ele o único a conseguir aceder às mesmas.

Após este *setup* inicial, para qualquer tipo de comandos que necessite do servidor, um *Cliente1*, por exemplo, comunica através do `main_fifo`, sendo este um FIFO restrito apenas a clientes autenticados. Este envia então os pedidos e, tal como acontece na ativação de um utilizador, sempre que este precisa de uma resposta do servidor é criado um FIFO temporário para esse efeito.

Existem situações nais quais o utilizador não necessita de comunicar com o servidor, pelo que escreve diretamente nas pastas em que tem permissão. Como por exemplo, quando precisa de escrever na pasta `sent/`, como tem permissões sobre a mesma, escreve diretamente sem recorrer ao servidor.

3. Programas

Foram desenvolvidos os seguintes programas executáveis:

- `server`
- `concordia-ativar`
- `concordia-desativar`
- `concordia-enviar`
- `concordia-ler`
- `concordia-responder`
- `concordia-remover`
- `concordia-listar`
- `concordia-grupo-criar`
- `concordia-grupo-remover`
- `concordia-grupo-destinario-adicionar`
- `concordia-grupo-destinario-remover`
- `concordia-grupo-listar`

3.1. `server`

Começando pelo nosso programa `server`. Este é responsável por atender os pedidos de clientes, tal como lidar com a gestão de utilizadores, mensagens e grupos.

Como mencionado previamente, o nosso servidor comunica através de FIFOs, estes servem para receber pedidos, assim como enviar estados ou outra informação útil.

Este, juntamente com os programas `concordia-ativar`, `concordia-desativar`, `concordia-grupo-criar` e `concordia-grupo-remover`, irão controlar todas as pastas relativamente a grupos e utilizadores no sistema Linux.

De maneira a gerir as mensagens, implementamos uma *Queue*, onde estas são guardadas em memória até um utilizador pedir os seus emails utilizando `concordia-listar`. Desta maneira garantimos que o servidor nunca necessita de permissões para as pastas `sent` e `received` do utilizador, assim evitando o acesso a estas e não as pondo em causa caso o servidor seja comprometido.

3.2. `concordia-ativar`

O processo de ativar um utilizador ao serviço implica a criação das estruturas fundamentais que possibilitam a interação do utilizador com o sistema, no caso as pastas onde serão armazenadas as mensagens enviadas e recebidas.

Ao receber a solicitação de ativar um novo utilizador, o serviço inicia o processo de criação das estruturas necessárias de acordo com os requisitos estabelecidos, criando então a diretoria para o utilizador e dando-lhe as devidas permissões de escrita e leitura para a mesma. Após a criação da diretoria, o utilizador, cria nesta as pastas de mensagens enviadas e recebidas, diretorias estas que só o utilizador tem permissões de escrita e leitura.

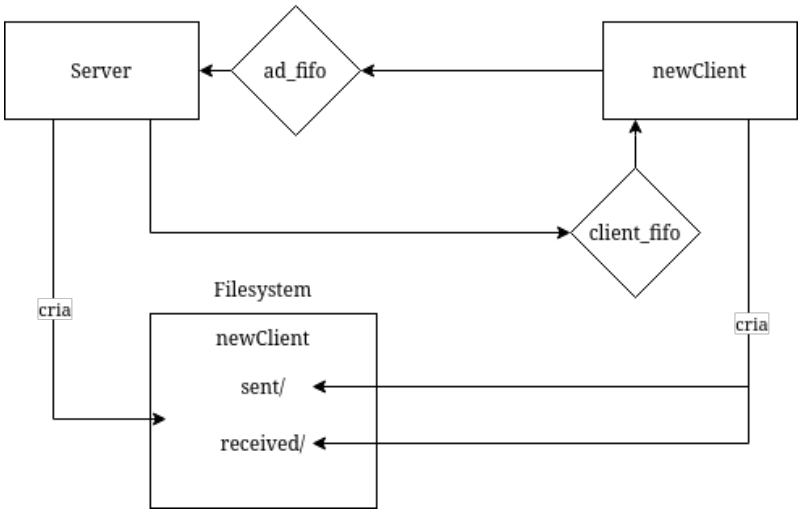

Figura 2: `concordia-ativar`

3.3. `concordia-desativar`

O processo de desativar um utilizador ao serviço implica a remoção das estruturas fundamentais que possibilitam a interação do utilizador com o sistema, no caso as pastas onde serão armazenadas as mensagens enviadas e recebidas.

Desta forma, tal como para o `concordia-ativar`, vai ser necessário uma comunicação entre o servidor e o cliente, porém esta será um bocado diferente. Começamos então pelo utilizador a remover as pastas de `sent/` e `received/`. Após isso, o utilizador informa o servidor e este por sua vez apaga a diretoria do utilizador.

3.4. `concordia-enviar`

O processo de enviar uma mensagem para um destinatário dentro do serviço é uma operação fundamental para a comunicação eficaz entre utilizadores. Após a composição da mensagem pelo remetente, a mesma é encaminhada para o destinatário desejado, que pode ser um utilizador individual ou um grupo de utilizadores.

Uma vez redigida, a mensagem é enviada e armazenada em uma *queue* de mensagens pendentes para entrega. Este sistema de *queue* permite que as mensagens sejam geridas de forma ordenada e processadas conforme a disponibilidade do serviço. Assim, mesmo em momentos de alta demanda, as mensagens são mantidas em *queue* e entregues aos destinatários assim que este as pedir, garantindo uma comunicação fluida e eficiente.

Ao utilizar este método de armazenamento em *queue*, conseguimos assegurar que o serviço nunca terá permissões de escrita perante as pastas de mensagens do utilizador. Desta forma, as mensagens são entregues de forma confiável e ordenada.

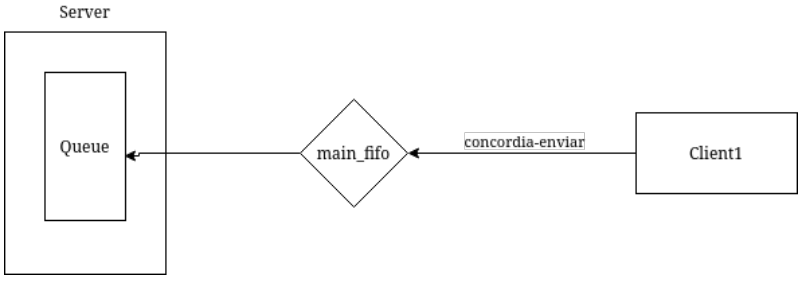

Figura 3: `concordia-enviar`

3.5. `concordia-ler`

A funcionalidade de ler o conteúdo de uma mensagem identificada pelo seu índice numérico é uma operação essencial dentro do serviço de mensagens. Esta, ao contrário dos programas prévios, não depende de qualquer interação com o servidor.

Deste modo, o cliente interage diretamente com os ficheiros que se encontrarem na pasta `received` do utilizador. Como o utilizador é *owner* de todos os ficheiros nesta pasta e todos os ficheiros nesta pasta tem um id, ao chamar `./concordia-ler <id>`, este programa irá aproveitar as permissões do utilizador para efetuar a leitura dos conteúdos do ficheiro e disponibilizar a informação ao utilizador.

3.6. `concordia-remover`

Quando um utilizador opta por apagar uma mensagem recebida, indicada pelo seu índice numérico, este, tal como para o `concordia-remover`, não terá que comunicar com o servidor.

Como mencionado previamente, o utilizador é *owner* dos seus ficheiros na pasta de `received`, deste modo este tem permissões sobre estes. Sendo assim, ao chamar `./concordia-remover` o programa utilizará a função `rm` do sistema no ficheiro, apagando-o.

3.7. `concordia-listar`

A funcionalidade de listar as mensagens é sem dúvida uma das mais importantes em todo o sistema.

O servidor, ao receber a solicitação para listar as novas mensagens, procura as mensagens na *queue*, onde estão armazenadas para processamento e entrega. Caso o sistema encontre mensagens, estas são retiradas da *queue*, enviadas para o utilizador pelo FIFO temporário criado pelo mesmo e escritas pelo utilizador na sua diretoria de mensagens recebidas.

Ao mesmo tempo que são retiradas da *queue* e enviadas ao utilizador, estas mensagens também serão listadas ao utilizador, demonstrando quem enviou a mensagem, quando foi enviada e o seu identificador único (índice numérico).

Caso o utilizador decida chamar este programa com a flag `[-a]`, para além de eventuais mensagens pendentes guardadas na *queue* do servidor, também serão listadas as mensagens já guardadas na pasta `received`.

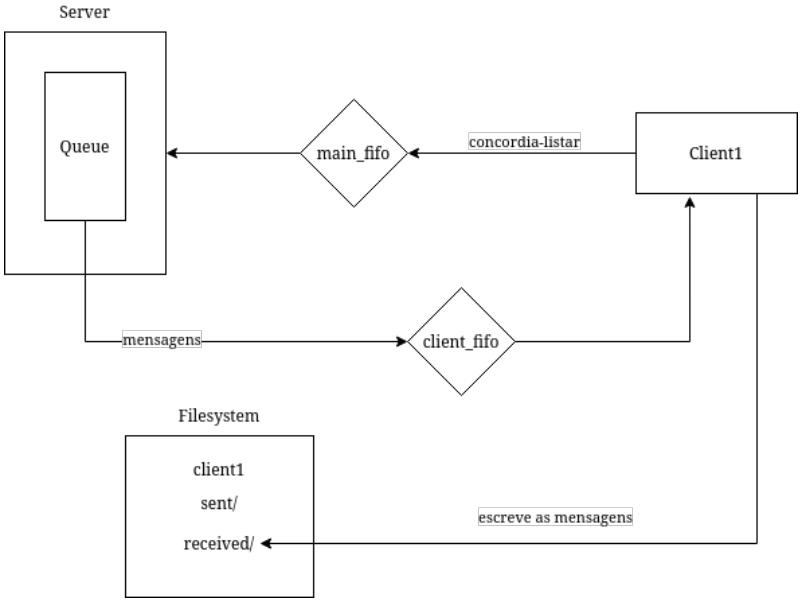

Figura 4: `concordia-listar`

3.8. `concordia-responder`

Apesar de não ser tão fundamental, implementamos também o método de responder a uma mensagem.

Tal como para o `concordia-enviar`, este programa terá que comunicar com o servidor de maneira a que este guarde a mensagem na *queue*. Porém este ao invés de receber um utilizador, recebe um *id* de uma mensagem. Tendo este *id*, conseguimos obter o utilizador a quem pretendemos responder lendo a mensagem.

Tenho o utilizador a quem pretendemos responder o processo é exatamente igual ao `concordia-enviar`.

3.9. `concordia-grupo-criar`

O `concordia-grupo-criar` é responsável por criar um grupo de utilizadores, criar as respetivas pastas para esse grupo e atribuir as permissões aos utilizadores que pertencem a esse grupo.

Este utiliza o comando **setfacl** para definir permissões específicas, para os utilizadores dentro desse grupo. Isso permite um controlo mais preciso sobre quem pode aceder e modificar os ficheiros e pastas dentro do grupo, garantindo a segurança e a integridade das mensagens. O uso do **setfacl** simplifica a administração de permissões, possibilitando uma gestão eficiente dos recursos compartilhados entre os utilizadores do grupo, **não sendo assim necessário atribuir permissões de root**.

3.10. `concordia-grupo-remover`

O `concordia-grupo-remover` é responsável por remover um grupo de utilizadores, apagar as pastas correspondentes e retirar as permissões a esse grupo.

3.11. `concordia-grupo-destinario-adicionar`

O `concordia-grupo-destinario-adicionar` é responsável por adicionar um utilizador a um grupo de utilizadores e atribuir as permissões a esse utilizador.

3.12. `concordia-grupo-destinario-remover`

O `concordia-grupo-destinario-remover` é responsável por remover um utilizador de um grupo de utilizadores e retirar as permissões a esse utilizador.

3.13. `concordia-grupo-listar`

O `concordia-grupo-listar` é responsável por listar todos os utilizadores de um grupo.

Para isto é usado o comando **getfacl** para obter informações sobre as permissões associadas aos ficheiros e pastas que pertencem ao grupo em questão. O **getfacl** é utilizado para exibir ACLs (Listas de Controlo de Acesso) em sistemas de ficheiros. Desta forma, o `concordia-grupo-listar` pode fornecer uma visão abrangente dos utilizadores que fazem parte do grupo, facilitando a administração e a gestão de permissões.

4. Daemon

Para a criação do *daemon* desenvolvemos um pequeno *script* em `bash` de forma a torná-lo mais fácil. Basicamente um ficheiro de serviço `systemd` é criado usando o comando `cat` combinado com `tee`. O conteúdo do arquivo é definido entre `<< EOF` e `EOF`, e é enviado para `/etc/systemd/system/` com o nome fornecido.

Neste ficheiro temos algumas informações sobre o nosso `.service`:

- **`Description:`** Uma descrição do serviço.
- **`ExecStart:`** Especifica o caminho completo para o executável que será iniciado como serviço. A opção `Restart=always` indica que o serviço será reiniciado automaticamente em caso de falha.
- **`WantedBy:`** Define a unidade de destino multi-utilizador como alvo desejado para a inicialização do serviço.

Após a utilização deste *script* é necessário executar o comando `systemctl daemon-reload`, para recarregar o `daemon` e aplicar as alterações.

Tendo feito estes passos todos, é possível então que o executável seja iniciado como um `daemon` e seja controlado pelo `systemd`.

Infelizmente, para a implementação do mesmo não tivemos o maior sucesso devido ao facto de estarmos ocasionalmente a usar alguns paths relativos e não dinâmicos na implementação.

5. Considerações Futuras

Para trabalhos futuros, uma consideração importante é a possibilidade de armazenar a *queue* de mensagens em disco e aplicar criptografia às mensagens para garantir a segurança e a privacidade dos dados dos utilizadores. Essa abordagem proporcionaria uma camada adicional de proteção aos dados sensíveis, mitigando possíveis vulnerabilidades de segurança e garantindo a confidencialidade das comunicações dentro do serviço.

Ao armazenar a fila de mensagens em disco, em vez de manter tudo na memória, poderíamos lidar com volumes maiores de dados de forma mais eficiente, permitindo a escalabilidade do sistema. Para além disso, a criptografia das mensagens antes do armazenamento garantiria que apenas os destinatários autorizados pudessem acessar e decifrar o conteúdo das mensagens, protegendo contra acessos não autorizados.

Adicionalmente, consideramos que seria interessante aplicar certificação das mensagens, desta forma conseguimos garantir a integridade das mesmas, retirando a possibilidade dos *owners* das mensagens as alterarem. Esta abordagem também nos permitiria implementar políticas de retenção de dados mais robustas, garantindo a conformidade com regulamentações de privacidade de dados e oferecendo aos utilizadores um maior controlo sobre o tempo de retenção de suas mensagens.

Portanto, como parte de um trabalho futuro, explorar a possibilidade de guardar a *queue* em disco e encriptar e certificar as mensagens seria uma medida importante para aprimorar a segurança e a eficiência do serviço de mensagens.

6. Problemas e Desafios

No contexto do projeto, a implementação do servidor foi um dos desafios mais significativos. Um servidor eficiente e seguro era essencial para garantir que o serviço de mensagens funcionasse de maneira confiável e segura.

Aspetos mais complicados de garantir no servidor:

- Garantir que este tinha as permissões mínimas necessárias para realizar suas tarefas, garantindo que não existiam vulnerabilidades que pudessem comprometer a segurança do sistema.
- O armazenamento e reencaminhamento de mensagens entre os utilizadores de forma eficiente e segura, garantindo que apenas os destinatários autorizados tenham acesso às mensagens.

7. Conclusão

Para concluir, percebemos que com este projeto aprendemos mais sobre os fundamentos da segurança de sistemas. Implementando funcionalidades como o envio de mensagens, a gestão de utilizadores e os grupos, e considerando aspetos de segurança como o controlo de acesso e a confidencialidade das mensagens.

Para além disso, a reflexão sobre as decisões arquiteturais e da implementação promoveram uma compreensão mais aprofundada dos desafios e das melhores formas de os ultrapassar.

No geral consideramos que este projeto é uma oportunidade valiosa para consolidar conhecimentos teóricos, desenvolver habilidades práticas e preparar os alunos para enfrentar desafios reais na área de segurança da informação.