Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Segurança de Sistemas Informáticos

Ano Letivo de 2023/2024

CONCÓRDIA

Serviço Local de Troca de Mensagens

António Filipe Castro Silva(a100533)
Flávio David Rodrigues Sousa(a100715)
Pedro Emanuel Organista Silva(a100745)

12 de maio de 2024

SSI

1 Introdução

Neste segundo trabalho prático foi nos pedido o desenvolvimento de um serviço de comunicação entre utilizadores locais de um sistema Linux. Para isso o nosso serviço resume-se à interação de um cliente com um servidor que se encontra sempre ativo e mantêm guardada todas as informações do que se passa neste serviço.

Do ponto de vista funcional, o serviço suporta o envio de mensagens para um utilizador, bem como a sua leitura pelo respetivo destinatário, funcionando de forma assíncrona, como um *mail*. Todos os utilizadores têm de ser ativados após inicialização pela primeira vez, sendo que após essa ativação, o servidor guarda num ficheiro todos os utilizadores que já estão ativos a partir do seu ID.

Cada cliente que já se encontra ativo tem a sua caixa de entrada, uma pasta com o seu ID, que somente ele consegue aceder de forma a ler as mensagens que recebe. Além de poder se ativar/desativar e enviar e ler as suas mensagens, este pode listar todas as mensagens que recebeu, pois só assim sabe qual é o ID da mensagem que quer ler, consegue responder a uma mensagem em específico e apagar mensagens.

O nosso serviço também suporta a criação de grupos privados de conversação. Sendo assim, as pessoas podem criar e remover grupos e adicionar e remover pessoas deles. Acontece que somente o criador do grupo pode apagá-lo e gerir as pessoas que estão nele, mas obviamente os membros do grupo conseguem enviar mensagens e receber mensagens dentro do grupo, sendo que as mensagens são sempre enviadas para todos os membros e ficam guardadas numa pasta com o nome do grupo. De forma a facilitar a leitura de mensagens de um grupo, criamos um comando para ver somente as mensagens do grupo. Um utilizador que é removido do grupo perderá imediatamente acesso ao conjunto de mensagens prévias e futuramente trocadas no grupo. Qualquer pessoa pode listar os membros de um certo grupo, mesmo não pertencendo a este, conseguindo assim ver que não consegue mandar mensagens para ele.

Tivemos sempre em atenção os aspetos de segurança, fazendo com que as pessoas só tivessem acesso e pudessem escrever para pastas que têm permissões. O servidor é sempre corrido com **sudo** de forma a poder tratar da criação das pastas para cada utilizador/grupo e mostrar conteúdos que só um certo utilizador pode aceder.

De forma a facilitar a arquitetura da aplicação em termos de código, dividimos de alguma forma o tratamento do comando que recebe, as ativações/desativações e envio/remoção de mensagens e o tratamento dos grupos dos utilizadores individuais.

2 Arquitetura Funcional do Serviço

Para inicializar o nosso serviço começa-se por abrir o servidor. Ao fazer isso ele vai criar dois **FIFOs** por onde vão passar as comunicações entre o cliente e o servidor. Após isso ele fica em *loop* à espera de comandos para mandar para o **Handler** a partir da função **handle_command**.

Após termos aberto o servidor, podemos finalmente abrir um cliente depois de termos feito **"su userX"** para usarmos outro utilizador. Este cliente inicialmente vai começar desativado e tem de enviar o comando **"concordia-ativar"** para o servidor o adicionar à lista de utilizadores ativados e autorizados a utilizar o serviço. Todos esses clientes vão ficar guardados num ficheiro chamado **"active.txt"** que guarda os ID's desses utilizadores. Além disso, o servidor trata da criação da caixa de entrada desse novo cliente, utilizando o seu ID como nome da pasta. Se o utilizador quiser desativar-se, basta utilizar o comando oposto **"concordia-desativar"**, desativando as suas permissões de utilizar o serviço e eliminando a sua caixa de entrada e todo o seu conteúdo.

Cada comando que o cliente faz manda uma certa mensagem (certo algumas exceções) para o servidor para este saber o que tem de fazer e depois envia esse comando e o seu conteúdo adicional (por exemplo, mensagem escrita) para o **Handler**. Vamos apresentar uma tabela na próxima página para explicar cada comando, o que se envia para o servidor, o que este faz com ela e que ficheiro trata dela posteriormente.

Conseguimos compreender então que a arquitetura deste serviço resume-se a: um cliente que vai enviando pedidos a um servidor e um servidor que fica à escuta desses pedidos. O servidor ao receber os pedidos vai mandá-los para o Handler que depois trata-os de acordo com uma certa estrutura de acordo com o comando em si, pois o cliente já formatou a mensagem inicialmente para facilitar essa leitura. Se for com uma mensagem, tem de mandar o destinatário e a mensagem em si, se for para criar/remover um grupo tem de mandar o nome do grupo, se for para adicionar/remover um membro de um grupo tem de mandar o nome do utilizador e o nome do grupo, sendo que o servidor depois verifica nos pedidos do grupo, se é o administrador a fazer esses pedidos, pois caso não seja, estes serão recusados. Depois dependendo de ser algo para um utilizador individual ou mais focado para grupos, vai para as funções do Files ou do Groups.

Além da tabela dos comandos, deixamos também um diagrama que resume facilmente a arquitetura do serviço.

Comando	Mensagem	Explicação	Quem trata
concordia-ativar	“ativar”	Pede ao servidor para ativar este cliente	Files
concordia-desativar	“desativar”	Pede ao servidor para desativar este cliente	Files
concordia-enviar	“enviar”+mensagem	Enviar para o servidor o pedido de enviar mensagem com o destinatário e a mensagem	Files
concordia-listar	Não envia mensagem	Lista todas as mensagens de um cliente	Client
concordia-ler	Não envia mensagem	Lê uma mensagem específica a partir do seu ID (só funciona se for uma mensagem sua)	Client
concordia-responder	“enviar”+mensagem	Enviar para o servidor o pedido de enviar mensagem com o destinatário e a mensagem, utilizando a mensagem a quem está a responder para ir buscar o destinatário	Files
concordia-remover	“remove”	Envia para o servidor o pedido de remover uma mensagem, enviando o ID dela (só funciona se for uma mensagem sua)	Files
concordia-grupo-criar	“grupoc”	Envia para o servidor o pedido de criar um grupo, com o nome dele	Groups
concordia-grupo-remover	“grupodel”	Envia para o servidor o pedido de remover um grupo, com o nome dele (só funciona se for o administrador)	Groups
concordia-grupo-destinatario-adicionar	“grupoadd”	Envia para o servidor o pedido de adicionar um membro a um grupo, com o nome dele e do grupo (só funciona se for o administrador)	Groups
concordia-grupo-destinatario-remover	“gruporem”	Envia para o servidor o pedido de remover um membro a um grupo, com o nome dele e do grupo (só funciona se for o administrador)	Groups
concordia-grupo-listar	Não envia mensagem	Lista todas os membros de um grupo	Client
concordia-grupo-ler	Não envia mensagem	Lista todas as mensagens de um grupo (só funciona se for um membro do grupo)	Client
close	Não envia mensagem	Fecha o serviço para esse cliente	Client

Figura 2.1: Comandos implementados no serviço

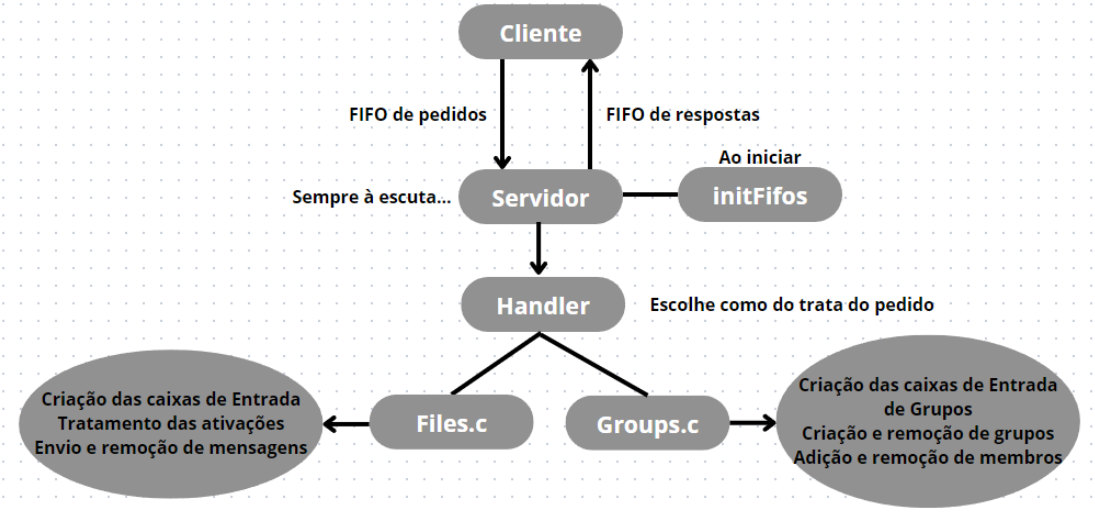

Figura 2.2: Arquitetura do serviço

3 Segurança do serviço e reflexão

Em termos de segurança de serviço, como temos vindo a mencionar, tivemos extrema atenção a forma como tratamos das permissões de cada utilizador com cada conteúdo de forma a que estes não conseguissem aceder a mensagens ou grupos que não deviam conseguir.

A forma com que tratamos disto foi ao criarmos as caixas de entrada, quer de cada utilizador, quer dos grupos, colocarmos logo as permissões para que só o *owner* delas consiga abrir ou as pessoas pertencentes ao grupo.

```
1     mkdir(path, S_IRWXU | S_IRGRP | S_IWGRP | S_IXGRP)
```

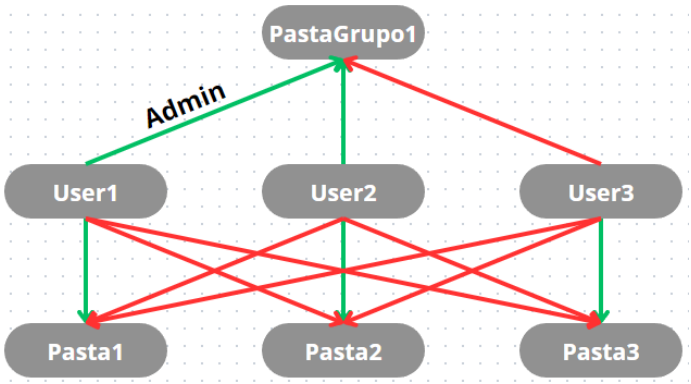

Figura 3.1: Exemplo de permissões

Como podemos ver pelo código da criação das pastas e utilizando a figura demonstramos como funcionam as permissões. Quando validamos os Users estes têm acesso à sua pasta individual e somente os Users do grupo têm acesso à pasta do grupo, logo, nesta figura, isso significa que somente o User1 e o User2 fazem parte dp Grupo1, logo só eles podem aceder à pasta do grupo. Como o User1 é o Admin, ele é o único que pode adicionar o User3 do grupo, dando-lhe acesso a essa pasta, e apagar o grupo, apagando essa pasta.

As pastas de cada User e Grupo, encontra-se dentro da pasta **data**, sendo que as pastas dos Users estão guardadas na pasta **users** e as pastas dos Grupos estão guardados na pasta **grupos**. Na pasta dos Users também é onde se guarda o ficheiro **active.txt** que indica ao servidor os clientes que estão ativados.

Outro parâmetro de segurança que já falamos brevemente são os administradores dos grupos. Quando um grupo é criado por um cliente este é designado como o administrador do grupo, isto

significa que este passa a ser o único membro desse grupo que pode apagá-lo, adicionar e remover membros nele. Sendo assim, qualquer membro que é adicionado ao grupo não pode adicionar outros membros e não consegue apagar o grupo em que foi adicionado.

Para executar a parte do serviço do servidor, tivemos sempre de corre-lo com **sudo**, pois este necessita de todas as permissões do sistema de forma a criar as pastas e mexer nas permissões delas. Depois do servidor criar as pastas, nós utilizamos o comando **chown** para alterar a *ownership* dessa pasta para o utilizador que se ativou ou para o grupo que foi criado.

Em termos de modularização, como se pode ver na pasta do **server**, dividimos o trabalho do servidor em 4 partes, tendo ainda nas **utils** mais duas funções (criação dos FIFOs e remoção de diretorias e os seus ficheiros) e os **paths** que auxiliam na criação dos FIFOs, das pastas onde se criam as caixas de entrada e para o ficheiro **"active.txt"**. O trabalho do servidor está dividido em:

- Server: criação dos FIFOs e escuta ativa dos comandos
- Handler: tratamento inicial dos comandos que o servidor recebe
- Files: criação das caixas de entrada dos utilizares, tratamento do estado de ativação de um utilizador(verificar, adicionar e remover) e adicionar e remover mensagens das caixas de entrada
- Groups: verificação do admin de um grupo, criação e remoção de um grupo e adição e remoção de um membro dum grupo.

Em termos de encapsulamento, o cliente e o servidor são totalmente independentes entre si no código, sendo que este só interagem entre si a partir dos FIFOs que são criados no servidor.

Para testar o nosso programa, enviamos mensagens entre vários utilizadores depois de estes serem ativados, conseguindo listá-las e lê-las com sucesso, utilizando os seus **Message_ID** (mid como temos no README do trabalho prático), criamos um grupo com dois utilizadores e vimos que estes conseguiam ambos ver as mensagens do grupo e quando um terceiro utilizador tentava aceder às mensagens do grupo, não conseguia, recebendo uma mensagem que se quisesse ver tinha de pedir autorização ao administrador. Além disso, ao remover o segundo utilizador, este também perdeu o acesso às mensagens.

Este trabalho prático também vem acompanhado com uma Makefile para facilitar a compilação do código. Após essa compilação basta executar o servidor numa consola utilizando o comando: **"sudo ./bin/server**. Após o servidor estar ligado e termos visto que ele se encontra "Esperando por pedidos...", podemos abrir outra consola, começando por fazer **"su userX"** para usarmos outro utilizador e depois basta fazer **"./bin/client** para correrrmos esse cliente. Na primeira vez, vai pedir para este ser ativado, mas depois se fecharmos e abrirmos o programa novamente, este já estará ativo até ordem contrária.

4 Conclusão

Depois de terminado este trabalho prático, conseguimos de melhor forma entender como as permissões determinam o acesso de certos utilizadores a certas pastas e como poderíamos utilizar isso para implementar um serviço de troca de mensagens entre utilizadores de um sistema.

Para a implementação deste trabalho, resolvemos tratar de tudo o que tinha a ver com utilizadores individuais e somente depois de termos a certeza que todas as permissões estavam direitas, pois apesar de este trabalho ter a ver com a troca de mensagens, a parte mais importante era ter a certeza que estavam a tratar das permissões corretamente, pois era o que se queria avaliar. Após implementarmos tudo dos utilizadores individuais, realmente procedemos à implementação dos grupos, que causou alguns problemas.

Um dos problemas que estávamos a ter era como íamos tratar das mensagens dos grupos, mais precisamente lê-las, visto que vários utilizadores as podem ver e teriam de ir para várias caixas de entrada. Além disso, como estes poderiam ser removidos dos grupos e é suposto perderem o acesso a essas mensagens, teríamos de estar a ver as mensagens da caixa da entrada do utilizador que vieram desse grupo para ele não as conseguir ver mais.

A forma que arranjamos para superar este problema foi criar uma caixa de entrada para cada grupo em que somente os membros desse grupo têm acesso. Foi por isso que criamos o comando **concordia-grupo-ler**, que um utilizador escreve especificando de que grupo ele quer ler as mensagens. Caso ele seja do grupo, consegue lê-las, caso contrário aparece um aviso que não pode e recomenda a pedir o acesso ao administrador.

Resumindo, achamos que conseguimos com sucesso implementar um serviço de comunicação entre utilizadores locais de um sistema Linux tendo sempre como primeira prioridade a utilização de permissões do sistema.