Universidade do Minho
Escola de Engenharia

Unidade Curricular de

Segurança de Sistemas Informáticos

Ano Letivo de 2023/2024

Serviço Local de Troca de Mensagens

Diogo Miranda (a100839)

João Rodrigues (a100598)

Sandra Cerqueira (a100681)

Maio, 2024

SSI

Índice

Introdução

Este projeto teve como objetivo desenvolver um serviço de mensagens assíncrono, denominado "Concordia", para utilizadores de sistemas Linux. A inspiração para este serviço provém da necessidade de proporcionar uma plataforma simples e robusta que facilite a comunicação entre os users sem necessitar de interações em tempo real, semelhante ao funcionamento do qmail.

O serviço "Concordia" foi projetado para operar integralmente através de comandos do terminal, permitindo enviar, listar, ler e responder mensagens, bem como gerenciar grupos de discussão. Dado o foco da unidade curricular na segurança, o projeto não se focou apenas na funcionalidade, mas também a segurança nas comunicações, adotando práticas de desenvolvimento seguro e mecanismos de controlo de acesso derivados diretamente das configurações do sistema operativo Linux.

Durante o desenvolvimento do "Concordia", várias decisões arquiteturais foram tomadas para maximizar a eficiência, segurança e usabilidade do serviço. A modularidade, encapsulamento e a integração com os sistemas de controlo de acesso do Linux foram prioritários, garantindo que apenas os usuários autorizados possam aceder às suas mensagens e grupos. Neste relatório iremos então detalhar o desenvolvimento do nosso sistema "Concordia", descrevendo a sua arquitetura funcional, as medidas de segurança implementadas, oferecendo uma reflexão sobre as escolhas arquiteturais realizadas.

Arquitetura Funcional

O serviço “Concordia” está organizado numa estrutura modular onde cada funcionalidade principal está implementada como um programa independente, permitindo uma gestão clara e facilitando a manutenção do código. A estrutura do sistema divide-se em diretorias e ficheiros principais da seguinte forma:
daemon: onde o nosso daemon fica a executar;

src: onde temos o código-fonte, dividido em vários módulos:

- **enviar.c, ler.c**, **listar.c**, **remover.c, responder.c** : Implementam as funcionalidades de enviar, ler, listar, remover e responder mensagens, respetivamente.
- **grupo.c:** gerência a criação e remoção de grupos de conversação e a adição ou remoção de membros.
- **Servidor.c**: Implementa o servidor que coordena a comunicação entre os clientes.

tmp: onde fica o FIFO do servidor

O módulo ‘servidor.c’ atua como o núcleo central do nosso serviço, gerindo todas as comunicações entre clientes. Ele é executado pelo nosso daemon, iniciando automaticamente no arranque do sistema, para garantir a disponibilidade contínua do sistema. Este módulo é responsável por “ouvir” e processar comandos enviados através do seu FIFO “Server_fifo.fifo”.

Operação principal: Quando o servidor inicia, ele começa por criar o seu FIFO, e em seguida abre-o para leitura.

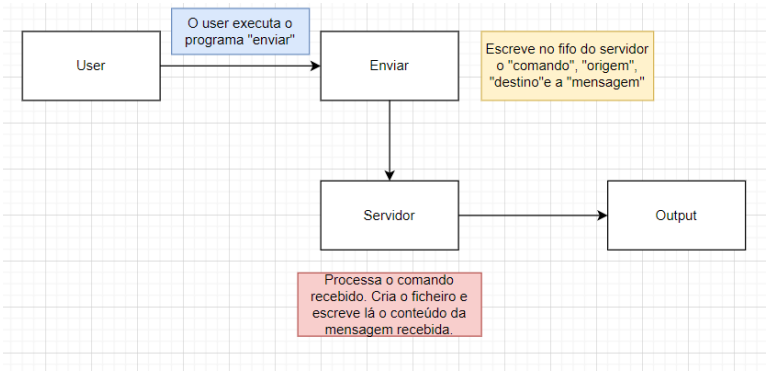
Processamento de Comandos: Os comandos são lidos do FIFO e processados num loop contínuo. Cada comando é analisado para determinar qual a sua função (como 'enviar' ou 'ler'), e as ações correspondentes são executadas:

- **Enviar**: No nosso sistema, o processo de envio de mensagens começa quando o user executa um comando como ./enviar <destinatário> <conteudo>, ativando o **enviar.c**. Este programa prepara e envia uma string formatada para o FIFO *'Server_fifo.fifo'*, do servidor (servidord.c), que então processa e armazena a mensagem. A segurança das mensagens é garantida pelo controlo rigoroso de acesso aos diretórios e arquivos onde as mensagens são armazenadas. Após extrair os dados necessários da mensagem recebida no seu FIFO, o servidor procede com o armazenamento da mensagem. Ele verifica a existência das diretorias necessárias na caixa de entrada do destinatário. Se não existirem, são criadas e configuradas com as permissões adequadas. Inicialmente, as diretórias são criadas com **permissões 0770**, permitindo que o proprietário (o destinatário da mensagem) tenha total controlo (leitura, escrita e execução), os users do grupo concordia têm controlo total também e os restantes users não têm qualquer permissão. O controlo total dado aos users do grupo concordia é apenas dado para configurar corretamente a pasta, sendo imediatamente retirado após a sua criação.

  A mensagem é então armazenada num arquivo dentro da diretoria "Novas Mensagens". O nome do arquivo é determinado pelo número sequencial de mensagens já presentes para evitar sobreposições. Cada arquivo de mensagem contém o nome do remetente, a data e hora do envio e o texto da mensagem.

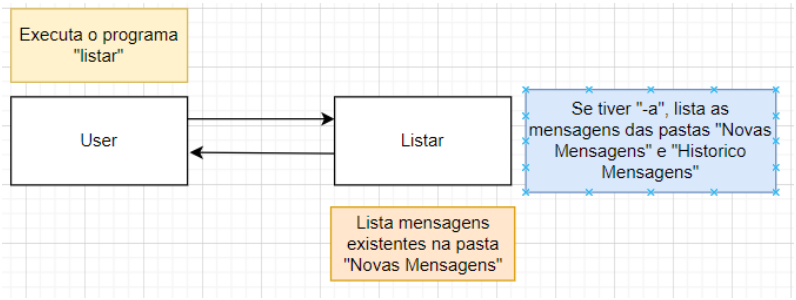
  Após criar e guardar o file de mensagem, o servidor ajusta as permissões do file para garantir que somente o destinatário especificado tenha acesso ao mesmo. Isto é feito através da função **chown()** para mudar a propriedade do file para o user destinatário, e aplicando **permissões 0700**, que permitem que apenas o proprietário do file tenha permissão de leitura, escrita e execução, assegurando assim a confidencialidade e integridade das comunicações dentro do sistema.

  Abaixo está representado, no diagrama, a interação entre o user e o servidor, no processo de enviar uma mensagem.



- **Listar**: a funcionalidade de listagem permite aos users visualizar as suas mensagens. Este processo começa com a execução do comando ./listar ou ./listar -a pelo user, utilizando o programa listar.c. Quando um user executa um destes comandos, o programa procede a identificar as diretorias relevantes na caixa de entrada do user, que está logado: a diretoria "Novas Mensagens" para mensagens não lidas e a diretoria "Histórico de Mensagens" para mensagens já lidas. Dependendo do comando específico executado (./listar para apenas novas mensagens ou ./listar -a para todas as mensagens), o programa acede à diretória apropriada. Posto isto ele abre a diretoria e começa a ler os files de mensagem um a um. Para cada file, o programa extrai informações essenciais como o ID da mensagem, que é inferido do nome do arquivo, o remetente, a data e a hora da mensagem e o tamanho do conteúdo da mensagem.

  As informações de cada mensagem são então apresentadas ao user de forma clara e legível. As mensagens são listadas com detalhes que incluem o ID, o remetente, a data e a hora, e o tamanho do conteúdo da mensagem. As mensagens que se encontram na pasta "Novas Mensagens" são mostradas primeiro. Se o comando ./listar -a for utilizado, as mensagens do "Histórico de Mensagens" são listadas depois, o proporciona uma visão completa das comunicações do user.
  Através do diagrama abaixo, conseguimos visualizar a interação entre o user e o programa listar.



- **Ler:** quando um user executa o comando *./listar <id_da_msg>*, ele está pedir para aceder especificamente à mensagem com o id fornecido, que se encontra na sua caixa de entrada. Ao receber o comando, o programa listar.c inicia verificando a validade dos argumentos fornecidos. O comando espera especificamente um id de mensagem. Se o número for omitido ou inserido incorretamente, o programa mostra uma mensagem de erro que indica o uso correto do comando.

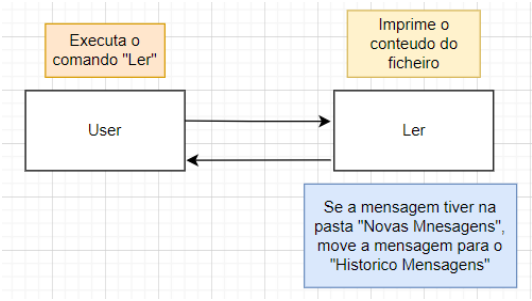
  O programa procede então para determinação da localização da mensagem dentro das diretorias de armazenamento de mensagens do user. Utilizando a variável de ambiente USER para identificar qual é a diretória específica do user, o programa acede à caixa de entrada. A procura da mensagem começa na diretoria "Novas Mensagens". Se a mensagem não for encontrada lá, o programa verifica a diretoria de "Histórico de Mensagens".

  Se a mensagem com ID estiver nas "Novas Mensagens", o programa mostra o seu conteúdo ao user e move a mensagem para o "Histórico de Mensagens", marcando-a assim como lida. Se estiver no "Histórico de Mensagens", a

mensagem é simplesmente lida. O file de mensagem é aberto, e os detalhes são mostrados.

Em casos onde a mensagem especificada não é encontrada em nenhuma das subdiretorias, o programa informa o usuário que a mensagem com o ID fornecido não existe. Isso assegura que o usuário esteja ciente de que o número inserido não corresponde a uma mensagem válida.

Através do diagrama abaixo, podemos visualizar a interação entre o user e o programa ler.



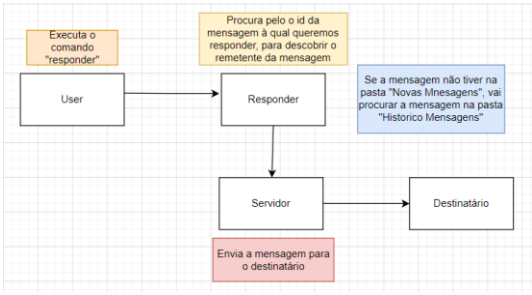
Responder: através do comando responder, o user pode responder a uma mensagem utilizando o seguinte comando *./responder <id_msg> <mensagem>*, com o, id_msg correspondendo ao id da mensagem à qual o user pretende responder.

O nosso programa inicia a sua operação procurando o id_msg na pasta “Novas Mensagens”. Se a mensagem não for encontrada, o programa tentará localizá-la na pasta “Histórico de Mensagens”. Uma vez encontrada a mensagem, conseguimos identificar o destinatário a quem pretendemos responder.

Tendo o obtido o remetente, o novo distinatário, passamos para a construção da mensagem a ser enviada, com osegundo argumento do nosso comando correspondendo ao conteudo da mensagem.

Desta forma, abrimos o fifo do servidor e eviamos a string que contém o comando, o remetente, o destinatário e a mensagem. Por fim, fechamos o fifo do servidor, e livertamos a memória alocada para construir a string.

Abaixo, podemos verificar o diagrama, que retrata a interação do user com o programa responder.

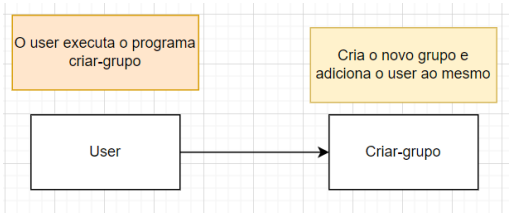


Criar Grupo: o nosso utilizador pode criar um grupo através do comando *./criar-grupo <nome_do_grupo>*.

O nosso programa começa por identificar o nome do utilizador que executou o comando. Em seguida, gera o comando *sudo groupadd <nome_do_grupo>*, criando um novo grupo com o nome fornecido pelo utilizador. Posteriormente, o programa adiciona o utilizador que criou o grupo ao mesmo, utilizando o comando *sudo usermod -aG <nome_do_grupo> <nome_do_utilizador>*. Desta forma, o utilizador é integrado no grupo que acabou de criar.

Durante a realização destes comandos, o nosso sistema verifica, em cada dos comandos se aconteceu algum erro durante a realização dos mesmos.

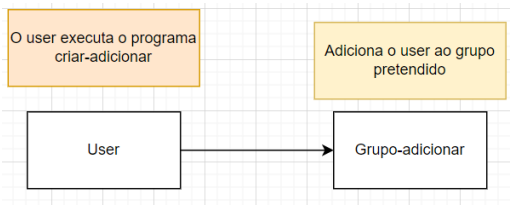
Através do diagrama abaixo, podemos verificar como funciona a iteração entre o user e o programa criar-grupo.



Grupo-adicionar: através deste programa, o nosso user pode adicionar um membro a um grupo já existente, para tal, basta utilizar o comando *./grupo-adicionar <nome_do_user> <nome_do_grupo>*.

O nosso programa vai começar por obter o nome do user a adicionar e o do grupo a qual vai ser adicionado, de modo a construir o comando *sudo usermod -aG <nome_do_grupo> <nome_do_user>*. Desta forma, conseguimos adicionar um novo membro ao grupo pretendido.

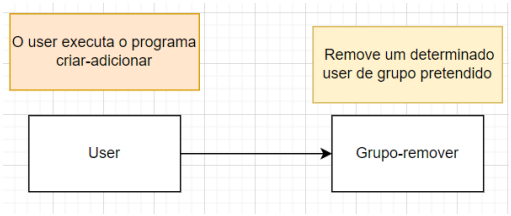
Através do diagrama abaixo, podemos verificar como funciona a iteração entre o user e o programa grupo-adicionar.



Grupo-remover: através deste programa, o nosso user pode remover um determinado membro de um determinado grupo, utilizando o seguinte comando *./grupo-remover <nome_do_user> <nome_do_grupo>*.

À semelhança do 'Grupo-adicionar', o nosso programa vai começar por obter o nome do user e do grupo no qual vamos efetuar uma operação. Mas, tal como o nome indica, o nosso programa vai ser responsável por remover um user de um determinado grupo. Para tal, ele vai utilizar os nomes fornecidos como argumentos para construir o comando, *sudo gpasswd -d <nome_do_user> <nome_do_grupo>*. Assim, o nosso user, consegue remover outro user de um determinado grupo.

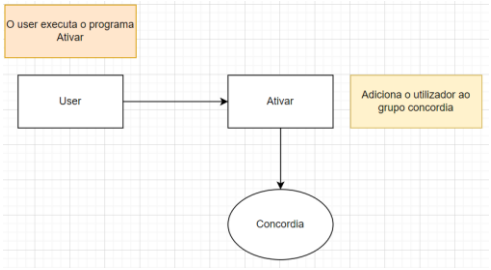
Através do diagrama abaixo, podemos verificar como funciona a iteração entre o user e o programa grupo-remover.



Ativar: para o nosso user poder interagir dentro do grupo 'concordia', primeiramente ele tem de entrar no mesmo. Para entrar dentro do grupo, ele terá de executar o comando *sudo ./ativar,* só desta forma o user será capaz de interagir dentro do nosso sistema e executar funções, como por exemplo, enviar mensagens, ler mensagens, etc.

Desta forma, quando o comando sudo é executado, o nosso programa vai construir o comando, *sudo usermod -aG concordia <nome_do_user>*, adicionando assim o user ao grupo 'concordia'.

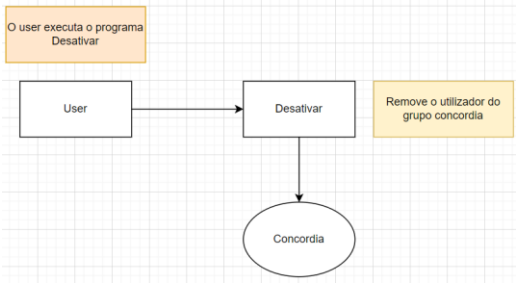
Através do diagrama abaixo, podemos verificar como funciona a iteração entre o user e o programa ativar.



Desativar: o programa desativar, ao contrário do ativar, é responsável por remover um user do grupo concordia, e consequentemente retirar as suas permissões para efetuar ações dentro desse grupo. Para tal o nosso user utiliza o comando sudo *./desativar*.

Desta forma, o nosso programa construi o comando *sudo gpasswd -d <nome_user> concordia*, removendo assim o user em questão do grupo.

Através do diagrama abaixo, podemos verificar como funciona a iteração entre o user e o programa desativar.



Resumo da Interoperabilidade e Mecanismos de Comunicação

FIFO do servidor: Utilizado como canal principal da comunicação entre os utilizadores e o servidor, facilitando a passagem dos comandos de forma eficiente e isolada do resto do sistema.

Estruturas de Dados e Dependências

Diretórias de Mensagens: Estruturas de diretórias são usadas para organizar mensagens em "Novas Mensagens" e "Histórico de Mensagens".

Arquivos de Mensagens: Cada mensagem é armazenada como um arquivo separado, permitindo acesso fácil e isolamento de dados entre diferentes usuários e sessões.

Decisões de Segurança tomadas

De seguida iremos descrever de forma geral as decisões de segurança tomadas, ainda que já tenham sido referidas algumas na secção anterior, com o objetivo de as sintetizar:

Criar as pastas

Durante a execução do nosso programa, caso a pasta “Caixa de Entrada” não exista, ou alguma das duas pastas nela contidas, “Novas Mensagens” e “Historico de Mensagens”, não existam, elas são criadas. Nesse momento são atribuídas as permissões 0700, de forma que apenas o próprio user possa aceder e/ou alterar alguma das pastas ou o servidor. Dessa forma garantimos a integridade das mensagens presentes nessas pastas, não podendo ser modificadas por mais ninguém.

Enviar mensagens

No processo de envio de uma mensagem, um arquivo para armazenar a mensagem, na “Caixa de Entrada” do destinatário, é criado. Esse arquivo é aberto com permissões de escrita e depois fechado, sendo no fim modificado o dono do ficheiro para o user destinatário, em vez de estar em nome do user que estiver a correr o servidor.

Reflexões

O nosso projeto, permite ao utilizador efetuar todas as funcionalidades pedidas no enunciado, como enviar mensagens, listar, ler e responder às mesmas, criar grupos, bem como adicionar novas pessoas aos mesmos. No entanto, identificamos uma limitação significativa na gestão de grupos. Da maneira que o código se encontra, não incluímos restrições adequadas para as operações de adição e remoção de membros, assim como a exclusão de grupos. Isto resultou na capacidade de qualquer user realizar estas ações, independentemente de serem ou não o criador do grupo. Esta falha leva a problemas de gestão de acesso e controlo dentro do sistema, comprometendo a segurança e a organização dos grupos. Temos consciência desta falha e com uma melhor gestão de tempo seria a primeira coisa a ser implementada futuramente.

Conclusão

Com este projeto conseguimos alcançar o objetivo de desenvolver um serviço de mensagens assíncrono para users dos sistemas Linux.

O serviço permite enviar, listar, ler e responder mensagens, além de gerir grupos de discussão através do terminal. A segurança tida como foco, com a tentativa de adoção de práticas de desenvolvimento seguro e mecanismos de controlo de acesso integrados às configurações do sistema operativo.

A arquitetura modular do sistema facilita a gestão e manutenção do código, dividido em diretorias e files principais que organizam cada funcionalidade de forma independente. Esta estrutura contribui para a eficiência, segurança e usabilidade do serviço, embora tenhamos identificado uma limitação crítica na gestão de grupos. A atual configuração permite que qualquer user adicione ou remova membros de um grupo ou até mesmo exclua um grupo, independentemente de ser o seu criador. Esta falha de controlo de acesso compromete a integridade e a privacidade do grupo.

Por fim, a experiência com este projeto não foi de todo a esperada, uma vez que o grupo enfrentou diversos desafios na atribuição das permissões, tal como o problema dos grupos. No entanto, apesar de necessitar de melhorias, o projeto tem potencial para uso prático e ajustes futuros.