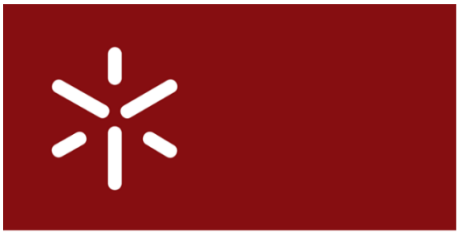

Relatório

Segurança de Sistemas Informáticos

2023/2024

Grupo 35

Nuno Aguiar a100480

Rui Cerqueira a100537

Guilherme Rio a100898

Índice

1. Introdução..... 3
2. Compilação..... 3
3. Arquitetura Funcional..... 4
3.1 Estruturas Principais:..... 4
3.2 Ficheiros secundários:..... 4
3.3 Execução do daemon.....5
3.4 Comunicação..... 5
3.5 Armazenamento das mensagens.....6
3.6 Verificação de mensagens de um cliente.....7
4. Segurança do Sistema..... 8
4.1 Users.....8
4.1.1 Ativar user:.....8
4.1.2 Desativar user:.....8
4.2 Grupos.....8
4.2.1 Criar grupo..... 8
4.2.2 Remover grupo..... 9
4.2.3 Adicionar/Remover Membro.....9
5. Conclusão e Trabalho futuro.....9

1. Introdução

No contexto da disciplina de Segurança de Sistemas, foi proposto o desenvolvimento do projeto "Concórdia", um serviço de comunicação entre os utilizadores locais de um sistema operativo Linux. Este serviço foi feito para oferecer de forma segura a comunicação, utilizando as funcionalidades do Linux, como o controlo de acesso e gestão de permissões.

O objetivo do projeto é desenvolver um serviço que permita o envio e recebimento de mensagens, semelhante ao email, e que inclui funcionalidades de gestão de utilizadores e grupo. É também necessário que o serviço consiga garantir a confidencialidade das mensagens.

2. Compilação

Para a compilação do projeto fizemos 2 escolhas, o local da pasta de instalação dos executáveis, que decidimos colocar na pasta */usr/local/bin* de modo a ser acessível a todos os utilizadores, colocando permissões de leitura a todos os utilizadores.

Para a comunicação por fifos decidimos criar também um local no sistema */var/lib/concordia/fifos* onde os utilizadores teriam todas as permissões exceto de remover fifos de outros utilizadores.

Por fim temos a localização da caixa de entrada do concordia no local */usr/share/concordiaINBOX/* onde os utilizadores poderão visualizar manualmente se pretenderem as suas respectivas mensagens.

De forma a se conseguir compilar o código é necessário fazer um *make install* para criar o executável do *daemon* e os restantes executáveis na pasta */bin* do projeto, para se obter os executáveis na pasta de acesso a todos os utilizadores é necessário realizar-se o comando *sudo make install*.

3. Arquitetura Funcional

De forma a estruturarmos o nosso projeto, decidimos dividi-lo em quatro estruturas principais.

3.1 Estruturas Principais:

Como mencionado no enunciado decidimos repartir os comandos disponíveis ao utilizador em três grupos(user, grupo, mensagem):

1. **concordia-user (concordia-user.c) -** Este é o grupo dos comandos relativos à criação e remoção de um utilizador.

2. **concordia-grupo (concordia-grupo.c) -** Estes são os comandos relativos à gestão de um grupo (criação, remoção, adicionar, remover e listar).

3. **concordia-mensagem (concordia-mensagem.c) -** Estes são todos os comandos relativos a ler e enviar mensagens, quer relativamente a utilizadores ou a grupos.

4. **daemon (daemon.c) -** O daemon funciona como o servidor, coordena a execução dos comandos e garante a segurança e integridade das mensagens trocadas.

De seguida, seguem-se todos os ficheiros a que o daemon e os concordia têm acesso, de forma a conseguir executar todos os comandos pretendidos pelo user.

3.2 Ficheiros secundários:

Ficheiros a apenas o *daemon* têm acesso:

1. **command_handler.c -** Este ficheiro é responsável por lidar com os comandos recebidos de forma a encaminhá-los para as devidas funções que os irão executar.

2. **user_commands.c, grupo_commands.c, message_commands.c -** Estes ficheiros contêm as implementações específicas dos comandos que se pretendem realizar, relativamente aos de user, os de grupo e aos das mensagens, respetivamente.

3. **utils.c -** Contêm funções utilitárias compartilhadas por diferentes partes do sistema.

4. **file_struct.c -** Contêm todas as funções necessárias que envolvem o tratamento dos ficheiros de mensagens requisitados pelo cliente.

Ficheiro compartilhado pelo *daemon* e pelos *concordia*.

5. **struct.h -** Contêm a estrutura dos pedidos trocados entre o cliente(*concordia*) e o servidor(*daemon*).

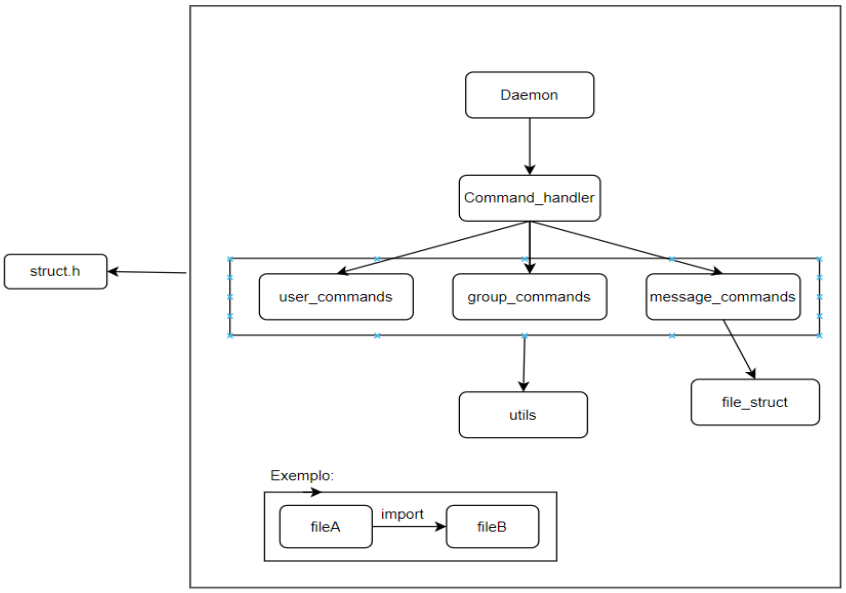

Figura 1: Arquitetura do daemon

3.3 Execução do *daemon*

De forma a aceitarmos múltiplos pedidos ao mesmo tempo, implementámos a criação de processos no *daemon*, sendo que cada vez que este recebe um pedido novo faz *fork()* de forma a que tenha um processo filho a atender as necessidades do cliente enquanto que o *daemon* principal continua a escutar por novos pedidos.

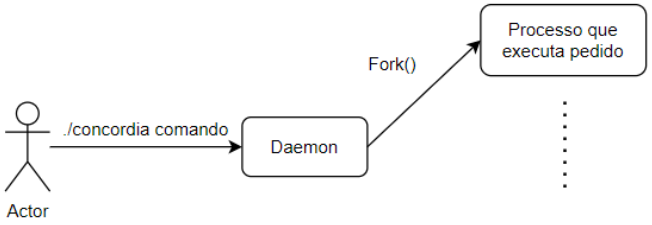

Figura 2: Representação da realização de um pedido

3.4 Comunicação

Para a comunicação entre os utilizadores e o daemon, é utilizado um fifo principal chamado concordia_fifo, para a troca de mensagens é utilizado uma estrutura *ConcordiaRequest,* esta estrutura leva uma flag a identificar qual o tipo da mensagem, o comando que indica qual o comando relacionado à mensagem, o user que identifica o user que enviou a mensagem, o destino que dependerá do tipo da mensagem, a mensagem, o campo *all_mid* que na mensagem de listar identifica se é para listar todas as mensagens ou apenas as não lidas, ou identifica o id da mensagem, e por fim temos o campo pid, que leva o identificador do processo que enviou a mensagem. O pid será posteriormente utilizado para realizar a comunicação servidor -> cliente, sendo criado um fifo individual para cada cliente utilizando o seu pid.

Figura 3: estrutura de requests

3.5 Armazenamento das mensagens

Relativamente ao armazenamento das mensagens foi estruturado um formato para o nome do ficheiro que irá armazenar a mensagem na diretoria do destinatário pretendido por quem a escreveu. Desta forma utilizamos os campos do nome do ficheiro para conseguir filtrar, gerir e identificar as mesmas.

O formato definido é o seguinte:

id;nomeDestino;nomeDeQuemEnviou;data|hora;tamanhoMensagem;bitLido;nRespostas;éUmaResposta

Assim, olhando para os campos separados pelos ‘;’ facilita-nos a maneira de interpretar a mensagem armazenada sendo que alguns destes campos irão ser interpretados no *file_struc.c* de forma diferente caso o destinatário seja um user ou um grupo.

Descrição dos campos:

- id: Identificador de uma mensagem.
- nomeDestino: Campo que contém o nome para onde esta mensagem foi enviada(seja ele um user ou um grupo).
- nomeDeQuemEnviou: Campo com o nome do utilizador que enviou a mensagem.

- data|hora: É a timestamp da data(dd-mm-aaaa) e hora(hh:mm:ss) exata da altura em que a mensagem foi enviada.
- tamanhoMensagem: Tamanho(bytes) que a mensagem escrita pelo utilizador ocupa.
- bitLido: Este campo funciona de forma diferente caso a mensagem esteja armazenada numa diretoria de grupo ou numa diretoria de user, no caso de ser para um utilizador o bitLido será 1(se este ficheiro já tiver sido lido) ou 0. Relativamente ao funcionamento deste campo em diretorias de grupos, sempre que um utilizador novo ler o ficheiro em causa é adicionado o seu nome ao final do ficheiro com um ‘;’ logo este campo conterá o número de bytes que foram adicionados ao final do ficheiro.
- nRespostas: Indica o número de respostas que um ficheiro já obteve.
- éUmaResposta: No caso deste campo ser 0 então indica que a mensagem dentro do ficheiro foi uma mensagem direta, caso seja uma resposta este campo terá o valor do id da mensagem que foi respondida.

3.6 Verificação de mensagens de um cliente

De forma ao cliente conseguir visualizar todas as mensagens a que têm acesso é necessário que o *daemon* comece por verificar todos os grupos com a função *getgrouplist,* equivalente ao comando *groups username* no terminal, a que o utilizador tem acesso. De seguida recolhemos e percorremos todas as diretorias que contêm os nomes na lista do que obtemos do *getgrouplist* e por fim criamos uma lista com o nome de todos os ficheiros nesta presente. Assim, sempre que necessitamos de mostrar a lista de mensagens de um utilizador ou chamamos um comando que identifique a mensagem pelo seu id é refeito este processo para garantirmos que a lista tenha sempre a mesma ordem.

4. Segurança do Sistema

De forma a manter a segurança neste projeto, tomamos a decisão de garantir as permissões de todos os ficheiros e diretorias ao daemon que corre como root do sistema, tendo total confiança neste serviço. Desta forma são criadas duas diretorias efetuando assim a divisão entre as diretorias de utilizadores e as diretorias de grupos.

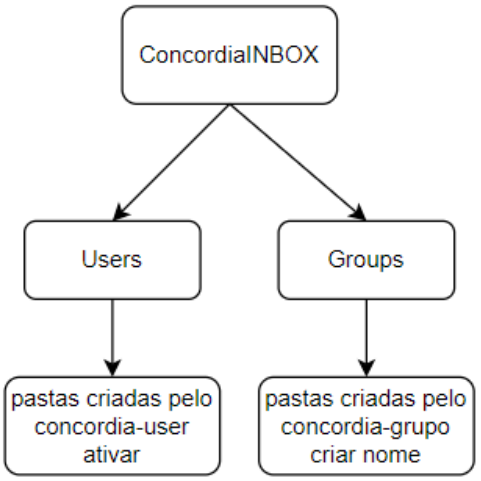

Figura 4: estrutura do armazenamento das diretorias

4.1 Users

4.1.1 Ativar user:

De forma a garantir a segurança dos ficheiros e o controlo da pasta dos users quando é requisitada a ativação da sua conta o daemon cria uma diretoria dentro da diretoria dos users com o nome do utilizador que requisitou a sua ativação, o dono desta será o daemon, oferecendo apenas as permissões necessárias para o utilizado conseguir aceder aos seus ficheiros mas não ter a possibilidade de os editar ou apagar.

4.1.2 Desativar user:

Para confirmar que apenas o utilizador que fez a ativação possa remover do sistema nós obtemos as permissões do utilizador no folder e verificamos se o utilizador tem permissões de leitura no ficheiro, como o dono do grupo é o único que as possui, caso isso seja verificado a diretoria é removida.

4.2 Grupos

Para garantir a segurança nos grupos dos utilizadores é garantido que cada grupo tenha a sua própria diretoria e um grupo de sistema associado para fazer a gestão dos ficheiros e o controlo de acesso estritos para proteger os dados do grupo. O daemon faz a configuração inicial necessária para tal.

4.2.1 Criar grupo

Começa então por ser criada a diretoria para o grupo criado, assim como um grupo com o mesmo nome no sistema linux. O grupo fica então gerido pelo root, e o grupo associado ao ficheiro. De seguida utilizamos ACLs para atribuir permissões de leitura e execução ao grupo, e não permitir outros utilizadores do sistema interagir.

Para as restantes mensagens precisávamos de ter uma maneira de identificar o criador do grupo, para isso na diretoria é criado um ficheiro “owner” que contém o nome do utilizador que criou o grupo, neste ficheiro são aplicadas as mesmas permissões que na diretoria. Por fim, adicionamos o utilizador que criou o grupo ao grupo do sistema.

Ou seja, em todas as mensagens os utilizadores poderão visualizar as mesmas mas não poderão efetuar quaisquer alterações nas mensagens recebidas ou enviadas.

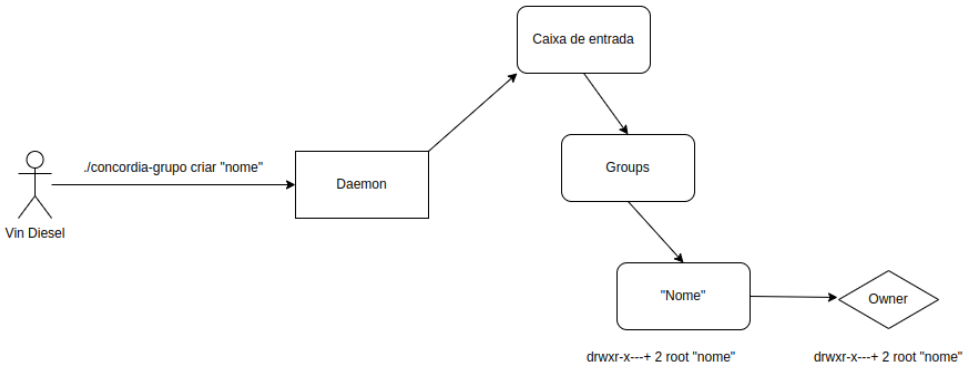

Figura 5: Demonstração da criação de um grupo

4.2.2 Remover grupo

Para a remoção de um grupo e verificar as permissões do utilizador que fez o pedido de remoção, verificamos com base no ficheiro owner do grupo para garantir que o utilizador tem as permissões necessárias, caso seja o dono será então feita a remoção da diretoria assim como do grupo do sistema. Caso não tenha as permissões necessárias será devolvida ao utilizador a informá-lo do mesmo.

4.2.3 Adicionar/Remover Membro

Para adicionar ou remover um membro é também verificada a ownership do utilizador que enviou a mensagem, caso seja validado o utilizador irá ser adicionado ao grupo do sistema e passará a ter acesso às mensagens trocadas por membros do grupo. Caso um utilizador seja removido este não conseguirá mais acessar quaisquer mensagens trocadas pelo grupo.

5. Conclusão e Trabalho futuro

Este trabalho não só consolidou os conhecimentos adquiridos em sala de aula, como também nos proporcionou uma oportunidade de os aplicar na prática.Ajudou-nos a analisar melhor a arquitetura de um projeto previamente à sua aplicação. No futuro gostaríamos de arranjar um método diferente na questão das permissões, sendo que atualmente estamos a confiar a 100% no *daemon* como utilizador root.