TÉCNICO LISBOA

PROGRAMAÇÃO DE SISTEMAS

Relatório Formal

Identificação dos Alunos:

Nome: Ion Ciobotari Número: 84075

Nome: Mihir Odhavji Número: 84151

Grupo: 2

Conteúdo

1. Introdução

O objetivo deste projeto é desenvolver um clipboard distribuído. Este clipboard distribuído deve ser capaz de aceitar um número ilimitado de aplicações e de outros clipboards (criados num mesmo computador ou noutros computadores).

Todos os clipboards criados e ligados ente si devem funcionar como backups. Caso um clipboard de um conjunto de clipboards interligados for abaixo, ao conectar-se novamente ao mesmo conjunto deve receber todos os dados mais recentes neles guardados.

Quanto às aplicações, estas possuem uma API predefinida que proporciona a capacidade de colar e copiar informação de clipboards ao qual a aplicação se encontra ligada. Para além disto, a API proporciona a capacidade de qualquer programador poder criar novas funcionalidades utilizando as funções já criadas.

2. Arquitetura do sistema

A arquitetura deste projeto encontra-se predefinida pelo responsável desta unidade curricular. A figura 1 demonstra a arquitetura utilizada para desenvolver este projeto.

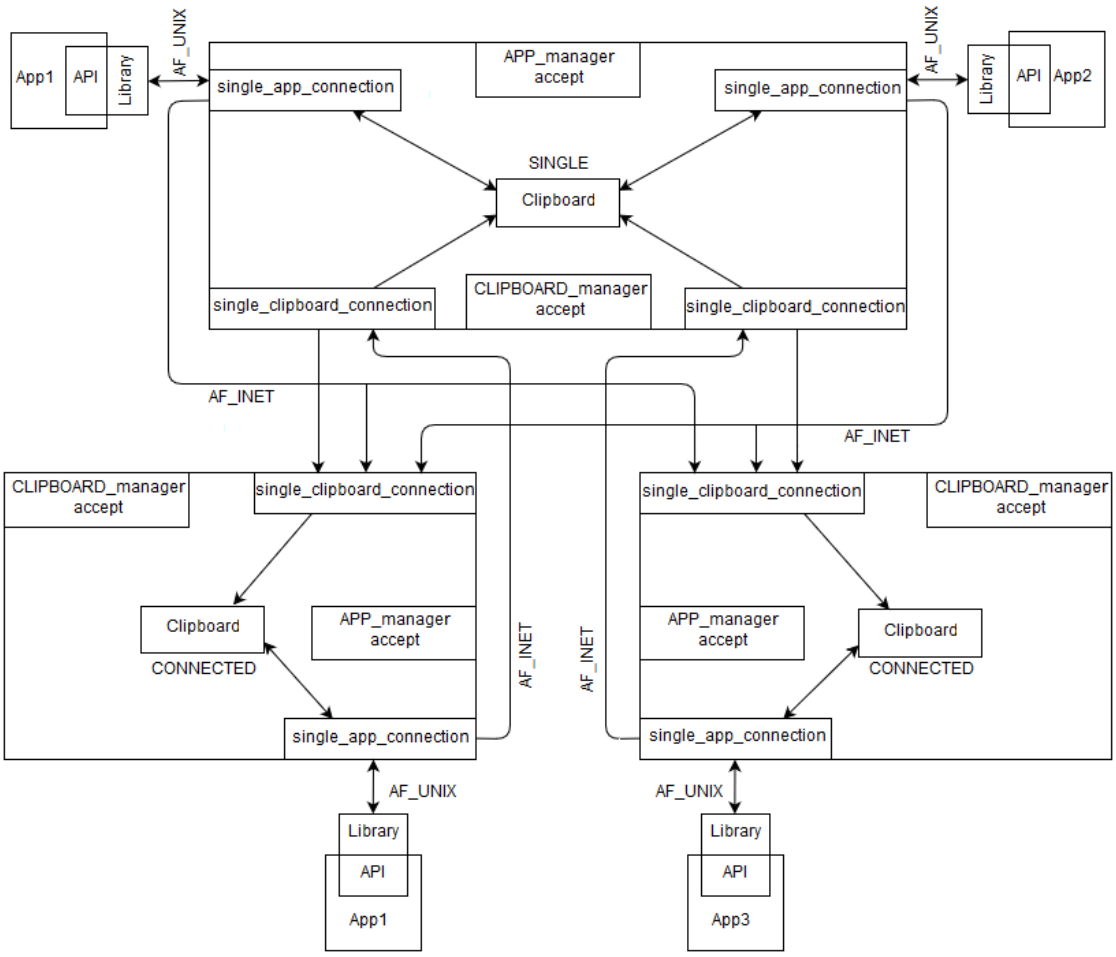


Figura 1- Arquitetura do projeto

Da figura anterior pode-se observar 3 componentes principais, API, Library e Clipboard. Estes componentes vão ter o seguinte funcionamento:

- **API** – inclui funções predefinidas que as aplicações podem utilizar para comunicar com o clipboard ao qual se conectou;
- **Library** – inclui todas as estruturas e funções que levam a implementar o API;
- **Clipboard** – componente responsável pelas conexões (via uso de threads) com aplicações e outros clipboards (criados num mesmo computador ou criados em computadores diferentes). Este componente é também responsável por receber os comandos vindos das aplicações, guardar dados e replicá-los a outros clipboards ao qual se encontra conectado.

Nota: no projeto submetido as componentes API e Library encontram-se num único ficheiro. O pretendido seria recorrer à implementação de bibliotecas dinâmicas de forma a indicar à API quais são as funções da Library a que tem direito de utilizar, contudo, esta implementação não foi possível de ser implementada.

2.1 Protocolo de Comunicação

De forma a iniciar o programa, o primeiro passo é ligar um clipboard. Para tal existem duas formas de o realizar:

- ./clip – comando que inicia um clipboard no modo single;
- ./clip -c X.X.X.X Y – comando que inica um clipboard no modo connected. X.X.X.X corresponde ao endereço IP no clipboard ao qual se pretende conectar e Y é porto pelo qual a comunicação se vai realizar. Para realizar uma conexão com um clipboard local, o endereço IP deve ser 0.0.0.0, 127.0.0.1 ou o próprio endereço do computador. O porto que se pode usar para criar conexões posteriores com outros clipboards tem de ser um que não esteja a ser utilizado. Um exemplo de comando para uma conexão entre dois clipboards locais seria ./clip -c 0.0.0.0 8100.

Para qualquer outro comando diferente dos demonstrados, o programa retorna erro com aviso de que o comando utilizado foi o incorreto.

Quando o clipboard é ligado no modo single, primeiro são inicializadas todas as regiões do clipboard, tal como as duas listas que guardam as identificações de todas as aplicações e clipboards que realizarão uma conexão. Para além disso, também são criadas duas threads (APP_manager e CLIPBOARD_manager) responsáveis por receber os pedidos de conexão das aplicações e dos clipboards, respetivamente.

Na thread APP_manager, as conexões com as aplicações são realizadas através de sockets do domínio AF_UNIX, uma vez que estas se encontram na mesma diretoria que o clipboard ao qual se quer conectar. Após realizada a conexão, na lista de contactos de aplicação é adicionado um novo nó com informação do file descriptor da aplicação e do identificador do socket pelo qual se vai realizar a comunicação entre a aplicação e clipboard. Depois da atualização da lista de contactos é criado uma nova thread (single_app_connection), responsável por receber e enviar toda a informação da aplicação ao qual está relacionado.

Na thread CLIPBOARD_manager, as conexões com os clipboards também são realizadas através de sockets, mas do domínio AF_INET. Este domínio possibilita a comunicação com outros clipboards em computadores diferentes, desde que se encontrem ligados à rede.

Tal como no APP_manager, após a conexão com um novo clipboard, este é adicionado na lista de contactos e de seguida é-lhe enviado o estado atual das regiões do clipboard e criado uma

nova thread (single_clipboard_connection), responsável por todas as comunicações com o novo clipboard e por pedidos de atualização ao clipboard de maior hierarquia.

O single_app_connection, como foi dito anteriormente, é a thread que gere todas as comunicações com uma única aplicação, com a capacidade de retirar informação diretamente das regiões do clipboard e enviar para as aplicações. Contudo, dependendo do modo de funcionamento do clipboard, esta thread tem o direito de atualizar as regiões do clipboard quando o clipboard é o de hierarquia mais alta, caso contrário, a nova informação recebida é enviada diretamente para o clipboard com hierarquia imediatamente superior. Quando a aplicação envia o sinal de fecho ou é fechada forçadamente, a thread também irá terminar. No caso de o clipboard ter terminado, a aplicação continua ativa apenas na tentativa de comunicar com o clipboard, é recebida uma notificação do encerramento do clipboard. Como já não é possível nenhuma comunicação, a aplicação irá encerrar.

O single_clipboard_connection é responsável por receber toda a comunicação do clipboard ao qual está ligado. Quando este recebe uma nova informação, caso o clipboard seja o de hierarquia mais alta, é realizada a atualização do clipboard e é enviado o novo estado para os restantes clipboards de mais baixa hierarquia. Caso o clipboard seja do modo connected, a informação recebida contém também um indicador se a informação recebida pode ser guardada ou não (GUARDA e NAOGUARDA). Quando o indicador é do tipo NAOGUARDA, a informação é enviada para o clipboard com hierarquia mais elevada, caso o indicador seja GUARDA, o estado do clipboard é atualizado e é enviado para todos os clipboards de hierarquia mais baixa com o mesmo indicador. No caso do clipboard ao qual está ligado terminar, a lista de contactos é atualizada de forma a ignorá-lo. No caso em que o clipboard que fechou é o de hierarquia mais alta é também feita a atualização de que o clipboard atual é o de maior hierarquia da árvore em que se encontra incluído. Continuamente na situação de encerramento de um clipboard, após as atualizações necessárias a thread responsável pela comunicação com tal clipboard também irá encerrar.

Quando o clipboard é ligado no modo connected, a única diferença para com o modo single é da criação de mais uma thread (single_clipboard_connection). Esta thread funciona como foi descrita anteriormente e neste caso o que receber é o estado das regiões do clipboard ao qual se liga. De seguida são criadas as thread APP_manager e CLIPBOARD_manager e o funcionamento é idêntico ao descrito anteriormente.

Nota: em anexo encontram-se os fluxogramas que descrevem melhor o funcionamento do programa.

2.2 Fluxo de Tratamento de Pedidos

As aplicações têm direito a utilizar 5 funções: clipboard_connect, clipboard_copy, clipboard_paste, clipboard_wait, clipboard_close. De forma a que o clipboard reconheça as diferentes funções que a aplicação pode realizar, quando a aplicação comunica com o clipboard esta também envia um indicador da função que está a realizar.

clipboard_connect – tal como o nome indica, esta função é responsável por enviar um pedido de conexão a um clipboard. Caso não exista clipboard, não é possível realizar a conexão e um erro é emitido.

clipboard_copy – esta função é usada quando se pretende enviar nova informação para o clipboard que se está ligado. Para enviar nova informação a função send é chamada duas vezes.

A primeira para enviar a região do clipboard para qual se quer guardar informação, o tamanho da mesma e a opção que se pretende fazer copy. No segundo é enviado o conteúdo que se pretende guardar.

clipboard_paste – ao contrário da função anterior, esta função tem como funcionalidade receber uma quantidade de bytes predefinidos pelo utilizador de uma dada região do clipboard. Primeiro é enviado ao clipboard a região de que se pretende obter informação. Para se receber a informação pretendida recorre-se à função recv. Se na primeira chamada desta função o clipboard indica que não existe informação na região pretendida, uma indicação para tal é emitida e termina a função, caso contrário a função recv é chamada continuamente até a quantidade de bytes pedidos pelo utilizador forem recebidos.

clipboard_wait – esta função é como clipboard_paste, com diferença de que o clipboard só envia informação da região pretendida quando esta for atualizada. Para que o clipboard tenha esta funcionalidade é usado a função pthread_cond_wait, desta forma a thread fica bloqueada nessa função até que haja uma nova alteração na região pretendida. De seguida o clipboard realiza o envio da informação pedida.

Clipboard_close – como o nome indica, a aplicação envia ao clipboard uma notificação que vai encerrar. Tendo recebido esta notificação a thread responsável pela aplicação também irá terminar.

3. Sincronização

Neste projeto, o clipboard local necessita de fazer várias operações ao mesmo tempo, e para tal, foi essencial a criação de várias threads. Ora algumas destas threads são threads equivalentes que comunicam com apps ou clipboards diferentes, e ao fazerem isso precisam de aceder às mesmas variáveis. Assim sendo, existe a possibilidade de threads diferentes tentarem aceder à mesma variável ou posição de memória ao mesmo tempo. Estas partes do código são chamadas de regiões críticas.

3.1 Identificação de regiões críticas

A região crítica mais evidente é o clipboard. Porém, se duas threads diferentes quiserem aceder a duas posições do clipboard diferentes, elas podem fazê-lo. Desde modo considerou-se que cada região do clipboard é uma região crítica independente.

A segunda região crítica identificada é a lista de contactos, ou seja, a lista onde foi guardado o file descriptor de todos os clipboards conectados. Isto é considerado uma região crítica visto que pode haver a probabilidade de duas threads tentarem enviar informação para um clipboard ao mesmo tempo utilizando o mesmo file descriptor.

Outra região crítica identificada é a lista com o file descriptor das aplicações. Considerou-se como região crítica, dado que existe a hipótese de mais de uma thread tentar aceder à cabeça da lista ao mesmo tempo.

A última região crítica identificada é a variável que guarda o modo de funcionamento do clipboard. Pode acontecer que a thread que trata da comunição com a hierarquia acima saia, mudando o modo de funcionamento e haver ao mesmo tempo uma thread que queira saber em que modo o clipboard se encontra, com a intenção de decidir se deve guardar a informação recebida, e enviar para a hierarquia abaixo, ou enviar para a hiearquia mais elevada.

3.2 Implementação de exclusão mútua

Para assegurar que apenas uma thread está dentro de uma região crítica temos de recorrer ao mecanismo de exclusão mútua. Nesta parte optou-se por usar read-write locks. Para justificar a nossa escolha temos primeiro de perceber como o read-write lock funciona.

Este lock pode ser usado em dois modos:

- Read lock – quando uma thread faz o read lock, o lock bloqueia threads que queiram escrever nessa região. Todavia este lock não bloqueia que outras threads leiam da mesma região, visto que uma leitura não vai alterar a informação lá guardada previamente. Diz-se que com o read lock é possível fazer leituras concorrentes.
- Write lock – quando uma thread faz write lock nenhuma outra thread pode aceder à aquela varíavel até que esta thread não faça unlock da variável. Um write lock bloqueia outros write lock, quer isto dizer que com write lock não é possível fazer escritas concorrentes, dado que, como é obvio uma escrita muda a informação da varíavel.

No projeto usaram-se read-write locks para limitar o acesso ao clipboard, quer para ler, quer para escrever.

Para o acesso às listas de contatos, o lógico seria usar read lock, visto que nós queremos só ler o file descriptor e não queremos alterá-lo. Contudo como queremos que apenas uma thread tenha acesso à lista de contatos de cada vez, optámos por usar um write lock. A linha de pensamento é análoga para a lista com o file descriptor das várias apps.

A opção de Wait requere que façamos uma espera dentro de uma zona ativa, mantendo o lock por um grande período de tempo, o que previne que outras thread acedam esta zona. Para resolver este problema escolhemos implementar o Wait com o uso de variáveis condicionais.

A função de pthread_cond_wait bloqueia a thread até receber sinal de que houve uma alteração na região do clipboard pretendida. Esta função só pode ser chamada depois da inserção de um lock no mutex. Esta função automaticamente desbloqueia o mutex enquanto está à espera do sinal. Este sinal pode ser emitido por duas funções: pthread_cond_signal e pthread_cond_broadcast. A diferença entre estes sinais deve-se ao facto de que o signal desbloqueia apenas um pthread_cond_wait enquanto que o broadcast desbloqueia vários. Estas duas funções também devem ser chamadas depois da inserção de um lock no mutex, e o lock deve ser retirado depois da sua chamada.

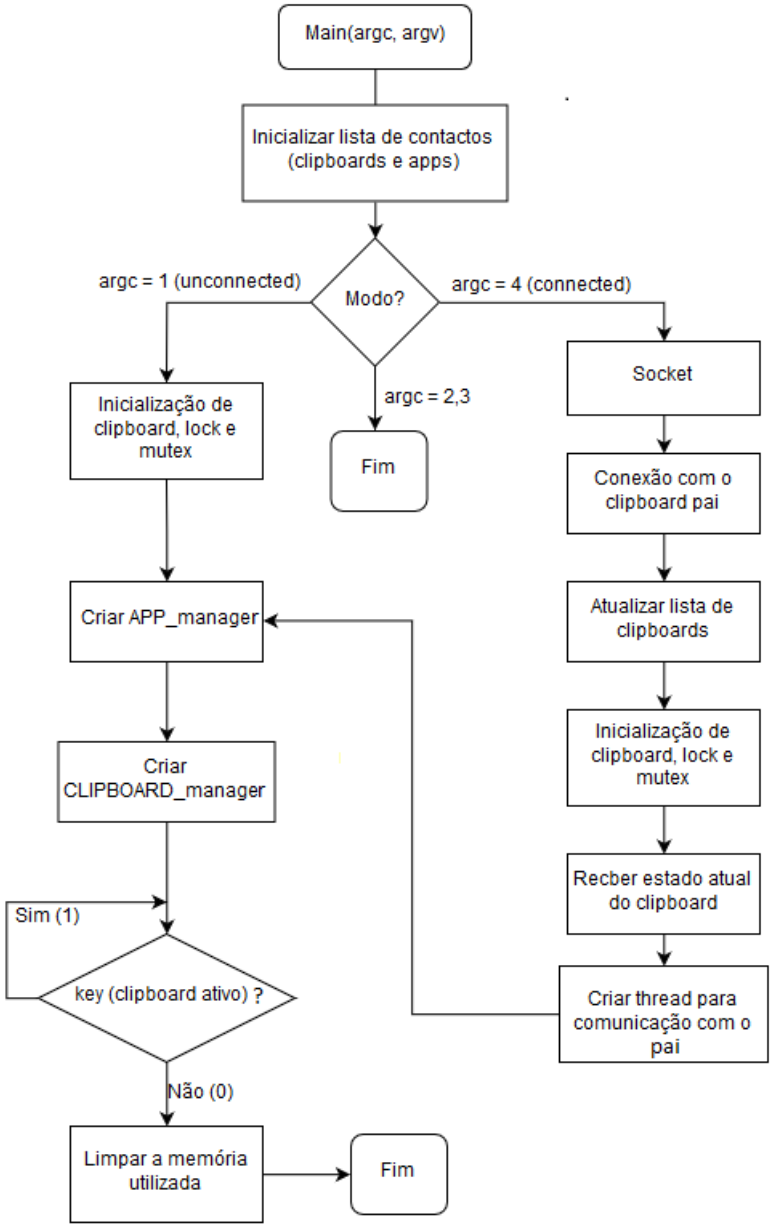
De modo a evitar o acesso múltiplo à variável que gere o modo de funcionamento do clipboard, também é necessária a implementação de write lock. Assim evitamos perdas de informação no caso de o clipboard mudar de modo de funcionamento no instante que se realiza um pedido de atualização ao clipboard de maior hierarquia. Contudo, no projeto submetido esta implementação não foi realizada.

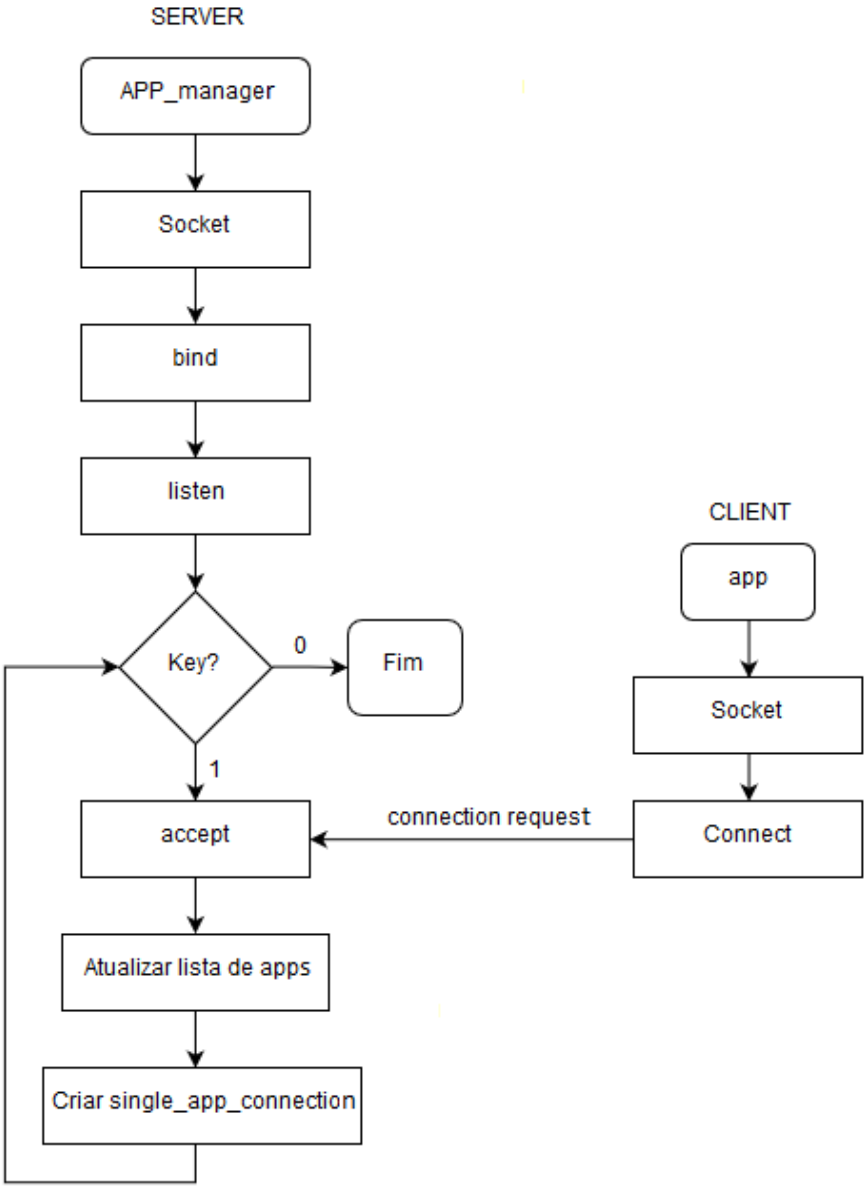
4. Gestão de recurso de tratamento de erros

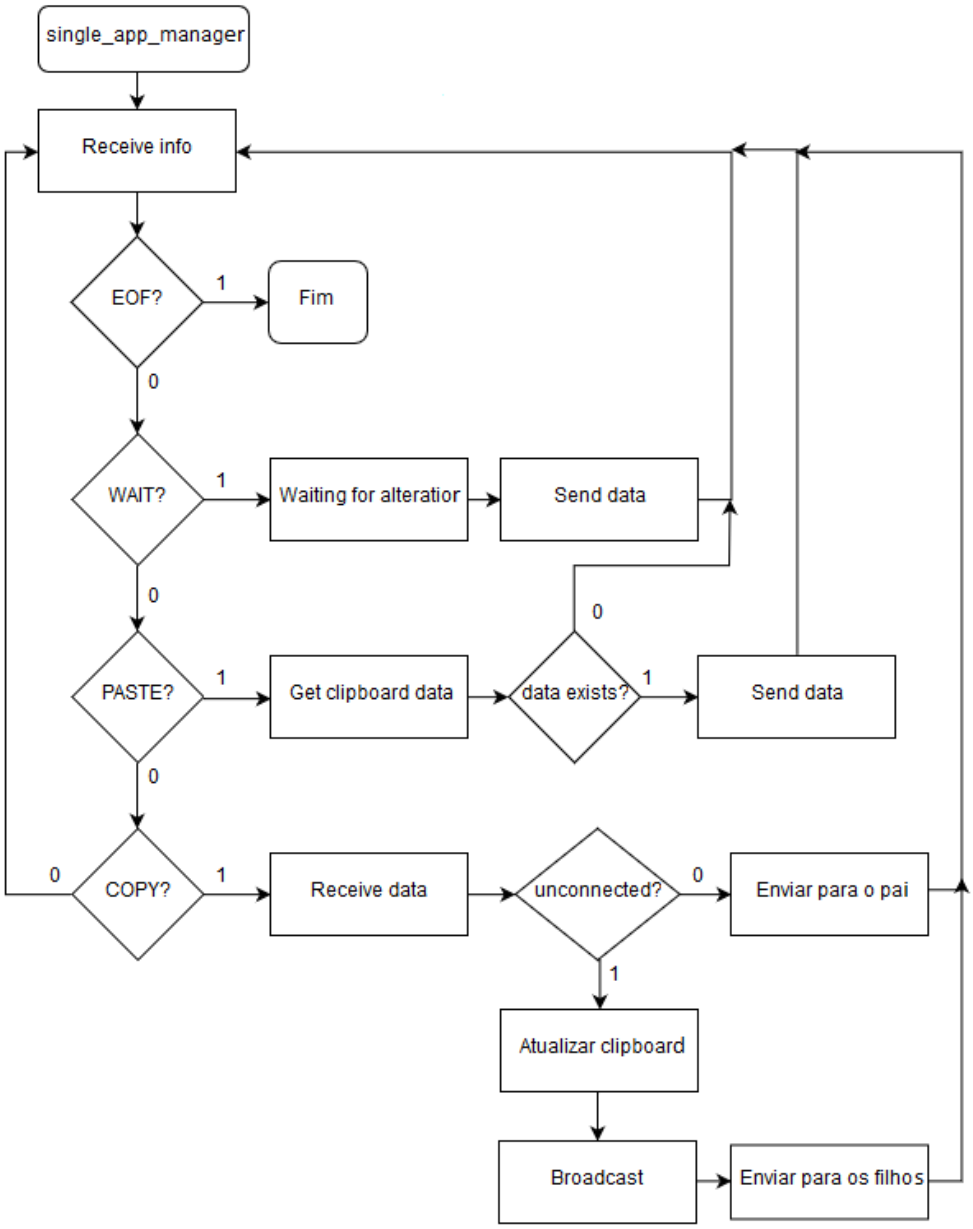
Ao longo de todo o programa são realizadas várias alocações de memória, envios e receções de informação e diversos locks e unlocks, ente outros. Sempre que é detetado um erro, por mínimo que seja, o programa começa um processo de libertar toda a memória que alocou desde que começou a correr. No início deste processo é libertado a memória das regiões do clipboard tal como a destruição das variáveis de cond, read-write lock e mutex. De seguida é libertada memória que foi utilizada para criar as listas de contactos e ao mesmo tempo, também é fechado o socket utilizado para comunicar com aplicações e clipboards.

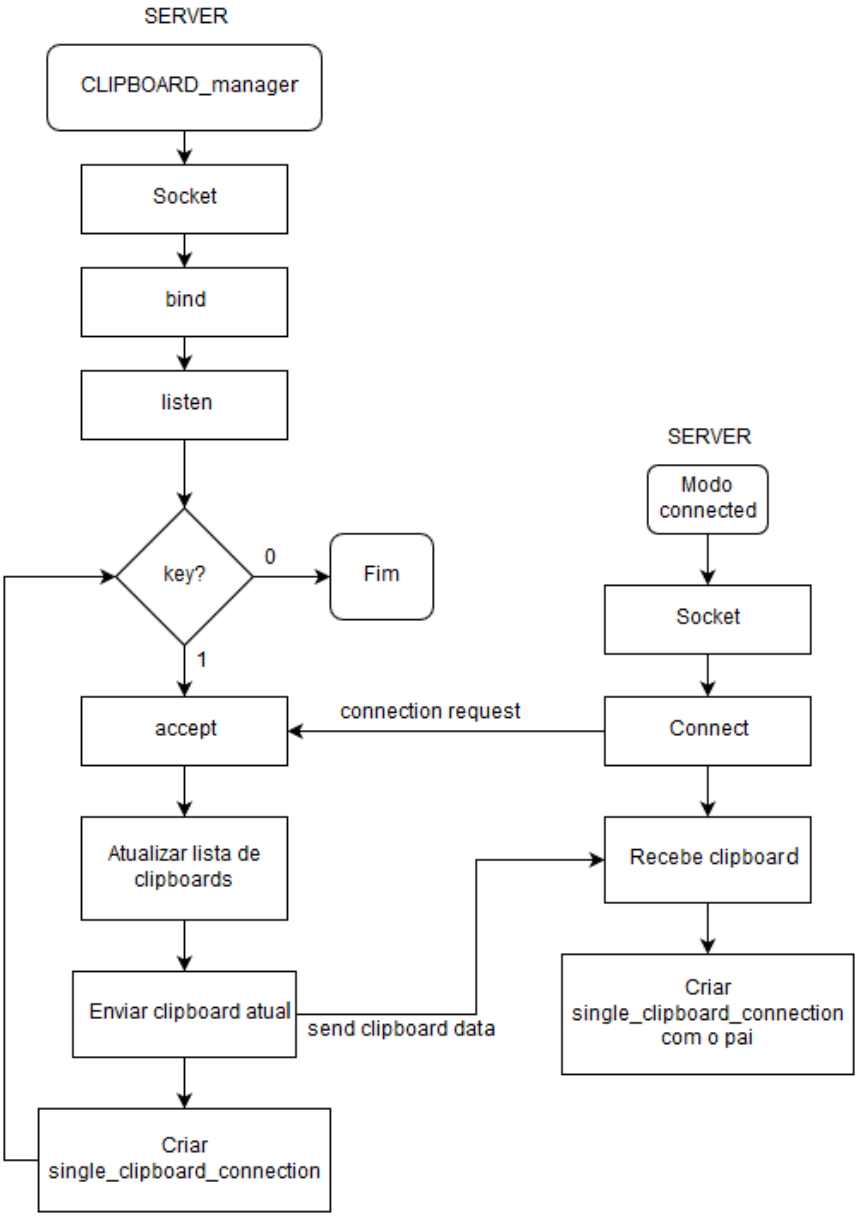
Acabado o processo de limpeza, o programa pode encerrar com segurança. No entanto, ao correr o pograma com o uso do software valgrind, este indica alguns erros de memória. Estes erros presume-se que sejam causados na tentativa de destruição das threads que foram criadas.

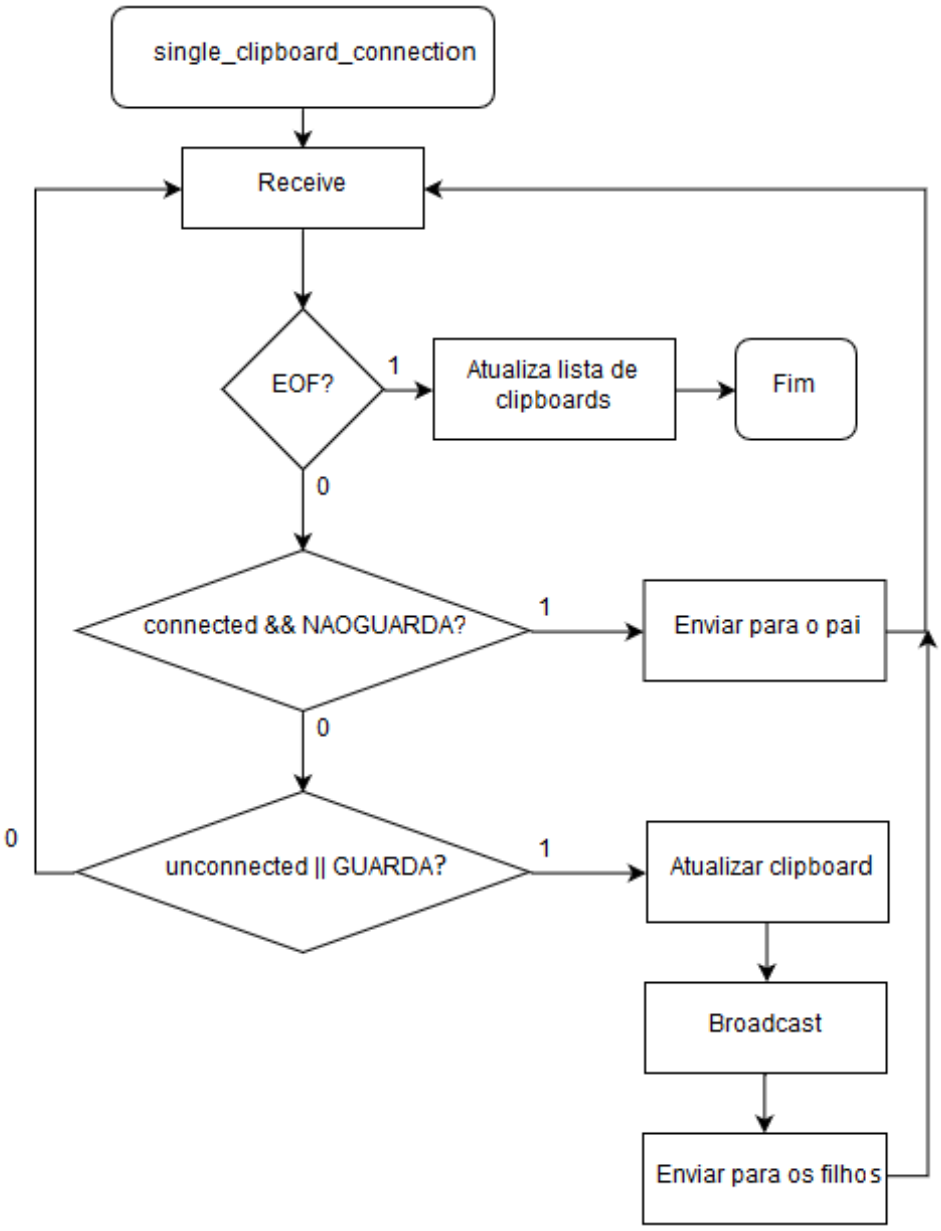
ANEXO


Main(argc, argv)
Inicializar lista de contactos (clipboards e apps)
Modo?
argc = 1 (unconnected)
argc = 4 (connected)
argc = 2,3
Fim
Inicialização de clipboard, lock e mutex
Criar APP_manager
Criar CLIPBOARD_manager
Sim (1)
key (clipboard ativo) ?
Não (0)
Limpar a memória utilizada
Fim
Socket
Conexão com o clipboard pai
Atualizar lista de clipboards
Inicialização de clipboard, lock e mutex
Recber estado atual do clipboard
Criar thread para comunicação com o pai


SERVER
APP_manager
Socket
bind
listen
Key?
0
Fim
1
accept
Atualizar lista de apps
Criar single_app_connection
CLIENT
app
Socket
connection request
Connect


single_app_manager
Receive info
EOF?
1
Fim
0
WAIT?
1
Waiting for alteratior
Send data
0
PASTE?
1
Get clipboard data
data exists?
0
1
Send data
0
COPY?
0
1
Receive data
unconnected?
0
Enviar para o pai
1
Atualizar clipboard
Broadcast
Enviar para os filhos


SERVER
CLIPBOARD_manager
Socket
bind
listen
key?
0
Fim
1
accept
Atualizar lista de clipboards
Enviar clipboard atual
Criar single_clipboard_connection
SERVER
Modo connected
Socket
connection request
Connect
Recebe clipboard
send clipboard data
Criar single_clipboard_connection com o pai


single_clipboard_connection
Receive
EOF?
1
Atualiza lista de clipboards
Fim
0
connected && NAOGUARDA?
1
Enviar para o pai
0
unconnected || GUARDA?
1
Atualizar clipboard
0
Broadcast
Enviar para os filhos