Instituto Superior Técnico

Engenharia Electrotécnica e de Computadores

Programação de Sistemas

2°Semestre 2017/2018

Projeto

84038 – Eduardo Melo
84037 – Eduardo Costa

Grupo: 51
Turno: Quarta-Feira 12:30 - 14:30
Docentes: João Nuno De Oliveira e Silva
Ricardo Miguel Ferreira Martins

07/06/2018

Conteúdo

1 Introdução

Foi-nos proposta a realização de uma aplicação que sirva como uma clipboard distribuída, de modo a que aplicações de clientes possam copiar informação para a clipboard (copy) e retirar informação desta (paste). Esta clipboard distribuída será corrida em diversas máquinas, sendo que o conteúdo destas terá que ser consistente entre todas as máquinas. De forma a realizar este projeto aplicámos os nossos conhecimentos de threads, sockets, sincronização e exclusão mútua, protocolos de comunicação e conhecimentos de cadeiras anteriores de programação.

2 Arquitetura do sistema

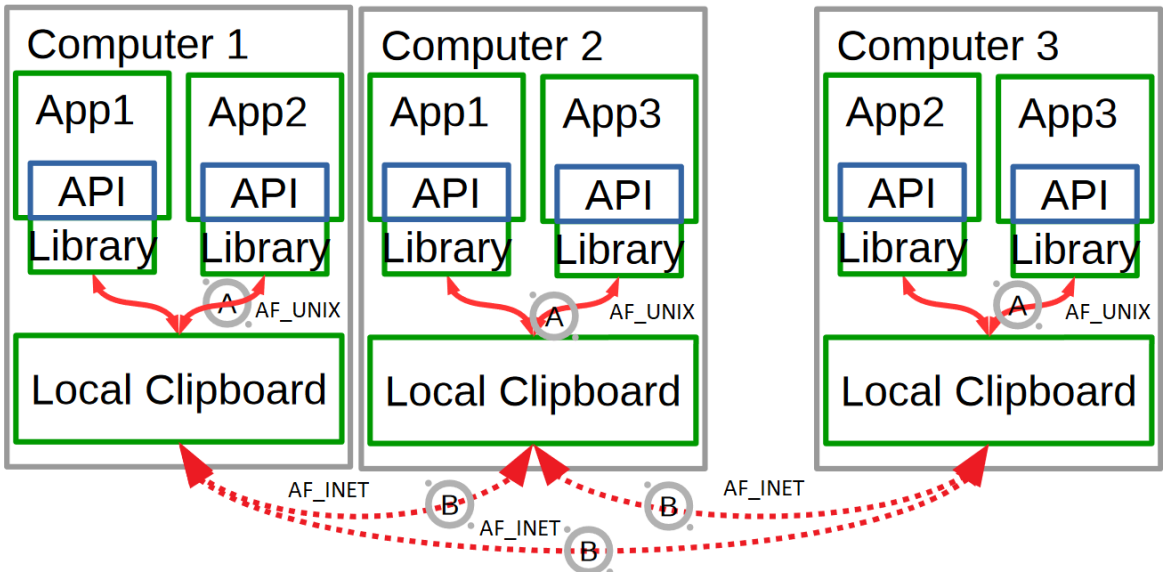

Figura 1: Arquitectura do sistema implementada

Na figura 1, encontram-se os diversos componentes implementados no sistema:

App Processo cliente que interage com a clipboard a partir de API Library;

API Library Conjunto de funções que servem de interface entre o local clipboard e outras aplicações;

Local Clipboard Conjunto de funções e variáveis que tratam dos acessos às diferentes regiões da clipboard, e que garantem que todas as clipboards de diferentes máquinas se encontram em sincronismo entre si.

2.1 Protocolo de Comunicação

Tanto na comunicação entre clipboards como na comunicação entre API Libraries e clipboards, o protocolo de comunicação utilizado é o User Datagram Protocol (UDP). Porém, como a comunicação entre o APP Library e clipboard é sempre efetuada na mesma máquina, o tipo de comunicação é socket AF_UNIX, enquanto que na comunicação entre clipboards, que se encontram sempre em máquinas diferentes, é do tipo socket AF_INET.

2.2 Fluxo de tratamento de pedidos

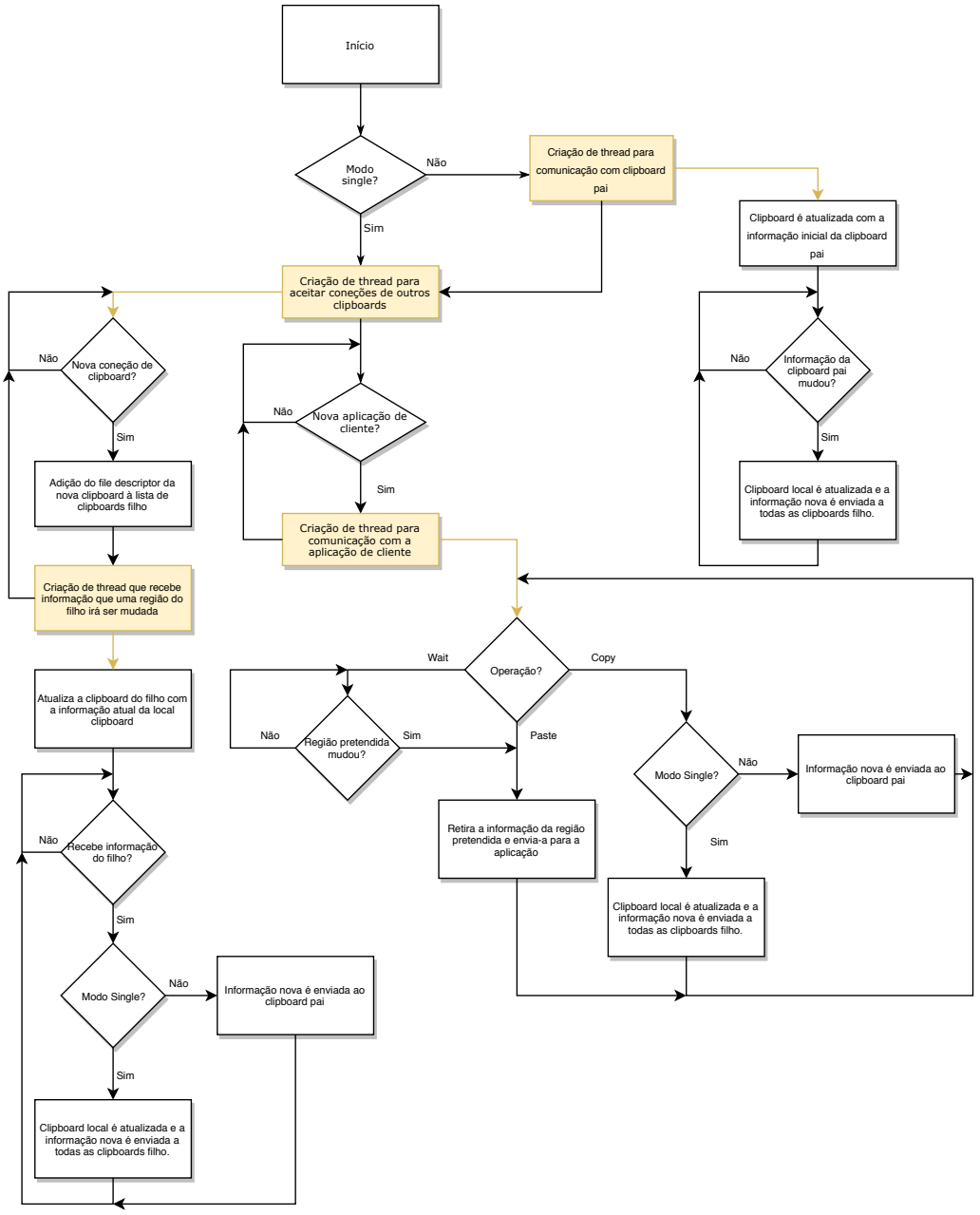

Figura 2: Fluxograma do sistema implementado

Na figura 2, é observável o fluxograma do sistema. São criadas threads em 4 alturas distintas (representadas a amarelo):

[Comunicação com clipboard pai]
Existe apenas 1 thread destas por clipboard local;
[Aceita coneções de outros clipboards]
Existe apenas 1 thread destas por clipboard local;
[Comunicação com um clipboard filho]
O número de threads é proporcional ao número de clipboards filhos conectados com a thread local;
[Comunicação com uma aplicação de cliente]
O número de threads é proporcional ao número de aplicações de clientes conectadas com a clipboard local.

Deste modo, o número de threads total do sistema é diretamente proporcional ao número de clipboards filhos e de aplicações de clientes conectadas.

Quando uma região de uma dada clipboard local é alterada, ou quando o clipboard local recebe informação de um dos filhos que uma região foi alterada, ela verifica qual é o seu modo (single ou connected), como ilustrado na figura 2. Se o clipboard local se encontrar em modo connected, este envia a informação para o clipboard pai, sem alterar a região em causa. Se o clipboard local se encontrar em modo single, a região em causa é alterada e a informação é enviada para todos os clipboards filhos, sendo estes posteriormente também alterados.

3 Sincronização

3.1 Identificação das regiões críticas

Existem 4 tipos de regiões críticas no programa:

[Criação do clipboard local]
Quando o clipboard local está a ser criado, a main thread é bloqueada (cond_wait), de forma a não conseguir criar nenhuma outra thread;
[Acesso à memória do clipboard local (read/write)]
Como podem existir várias threads a querer aceder ao clipboard local é necessário sincronizar os acessos;
[Acesso à lista de clipboards filhos conectados]
Como podem existir várias threads a querer aceder à lista de filhos conectados, é necessário sincronizar os acessos;
[Envio de informação ao clipboard pai]
Garante que a informação é enviada ao clipboard pai pela ordem correta.

3.2 Implementação de exclusão mútua

[Criação do clipboard local]
Foi utilizado mutex e uma variável de condição. A main thread fica condicionalmente a dormir até a thread que cria o clipboard local seja completamente atualizada, enviando posteriormente o sinal para a main thread acordar.

[Acesso à memória do clipboard local (read/write)]
Foi utlizado 1 rwlock por região da clipboard local. Estes garantem os acessos síncronos às diversas regiões do clipboard local.

[Acesso à lista de clipboards filhos conectados]
Foi utilizado 1 rwlock para a lista de clipboards filhos conectados ao clipboard local. Este garante o acesso síncrono à lista.

[Envio de informação ao clipboard pai]
Foi utilizado mutex que garante que só 1 thread está a mandar informação ao clipboard pai num dado momento.

Adicionalmente foram utilizados 1 mutex e 1 variável de condição por região da clipboard local, de forma a implementar a função wait. A variável de condição bloqueia a thread até que a região correspondente da clipboard local seja alterada. Depois de alterada, a thread é desbloqueada e realiza a operação de paste da região correspondente.
Todas as variáveis de sincronismo, o modo de funcionamento do clipboard local, a lista de clipboards filhos conectados ao clipboard local, o tamanho atual em bytes de cada região do clipboard local e o clipboard local são partilhados por todas as threads de uma dada máquina.

4 Gestão de recursos e tratamento de erros

Existe tratamento de erros ao longo do código. Abaixo apresentam-se os principais tipos de erros e o tratamento destes.

malloc()
Print de uma mensagem de erro para o stdout, libertação da memória alocada na thread corrente e pthread_exit().

erro de criação de socket e de accept()
Print de uma mensagem de erro para o stdout. Se o socket em questão conectar a clipboard local com o pai ou se der listen de conecções de aplicações de clientes, a memória alocada é libertada e a main thread é retornada.

erro de send() e receive()
Print de uma mensagem de erro para o stdout, libertação de memória, chamada de funções que garantem a replicação na árvore (se for necessário) e pthread_exit().

criação de threads
Print de uma mensagem de erro para o stdout e mudança do modo de execução caso seja necessário (de connected para single).

Quando uma thread é criada, dá-se pthread_detach() da mesma, de forma a que quando pthread_exit() é chamado a memória desta seja devolvida ao sistema operativo.

5 Critérios de avaliação

Consideramos que os critérios de avaliação encontram-se todos 100% completos, com excepção do tratamento correto de recursos, da sincronização eficiente e da estrutura do código.

[Tratamento correto de recursos]
Não foi implementado um action handler para o sinal sigint, e portanto não é libertada toda a memória do programa na saída deste.

[Sincronização eficiente]
Não conseguimos garantir que a nossa solução para as regiões críticas seja a mais eficiente possível.

[Estrutura do código]
O código implementado poderia estar mais organizado, e poderiam ter sido criadas funções que minimizassem a repetição de código.