CSaP Project: A AudioMessagge application

Database-Server,Client-Server approch

Fully writen in C (Tested on ArchLinux ,Pop OS 21.04,Parrot OS,kali linux) .

Parts:

Client:Interact with the user;

VDR: VoiceDataRepository,is aim is to store and update list of message of various user(his inbox) in an efficent way.

MDS:Main server, interact with the user for authentication and operate between Client and VDR as intermediary.

Features(Design Choices)

Use PortAudio library (insted of Alsamixer or Pulseaudio ones) for better inter-compatibility.

Auto-crop silence in the message by analyzing peek frequency.

Possibility of search for messages from a specific user, a date range and even the two things tougheter!

Possibility to send long audio without death times ,no time limit for audio rec has still reconized .

Socket operation are made atomical by design of specific function (no large buffers are used) for improve security.

Huge use of semaphore and parent-child processes(use POSIX semaphore insted of system five one)..

The project is highly scalable(you can add how much VDRs you want), but can still improve(es. Decentralized Mdss that can comunicate each other).

Every message is checked for corruption before every tranfer between VDR and MDS or MDS and CLIENT and vice versa.

possibility to remove all the inbox messages with one call

or to delate message one by one.

change password for logged users implemented.

some security checks in the calls to prevent attacks using a modded version of the source code .

portaudio already present and bundled you only need to install some audio related library .

80's style command line interface.

What happened if a part fail(a part die for some external reason)?

If mds fail the client is disconnected and vdr close the connection and waiting for the mds to reconnect

If a vdr fail the Mds will close the connection with the clients and die (no restart or use of goto to go before connection since is more easy to set up an auto restart system)

If a client fail well the socket for that client and child process in the mds are correctly closed and you can easily reconnect

Dependencies:

all the dependencies nedded to use a bundled version of portaudio

-ljack libjack-dev

sudo apt install libjack-dev

-lsndfile

sudo apt-get install libsndfile-dev

-lasound

sudo apt-get install libasound2-dev

How to Install?

an easy script is already here for debian users

How to test ?

there is a script for everything in this project

How to clean

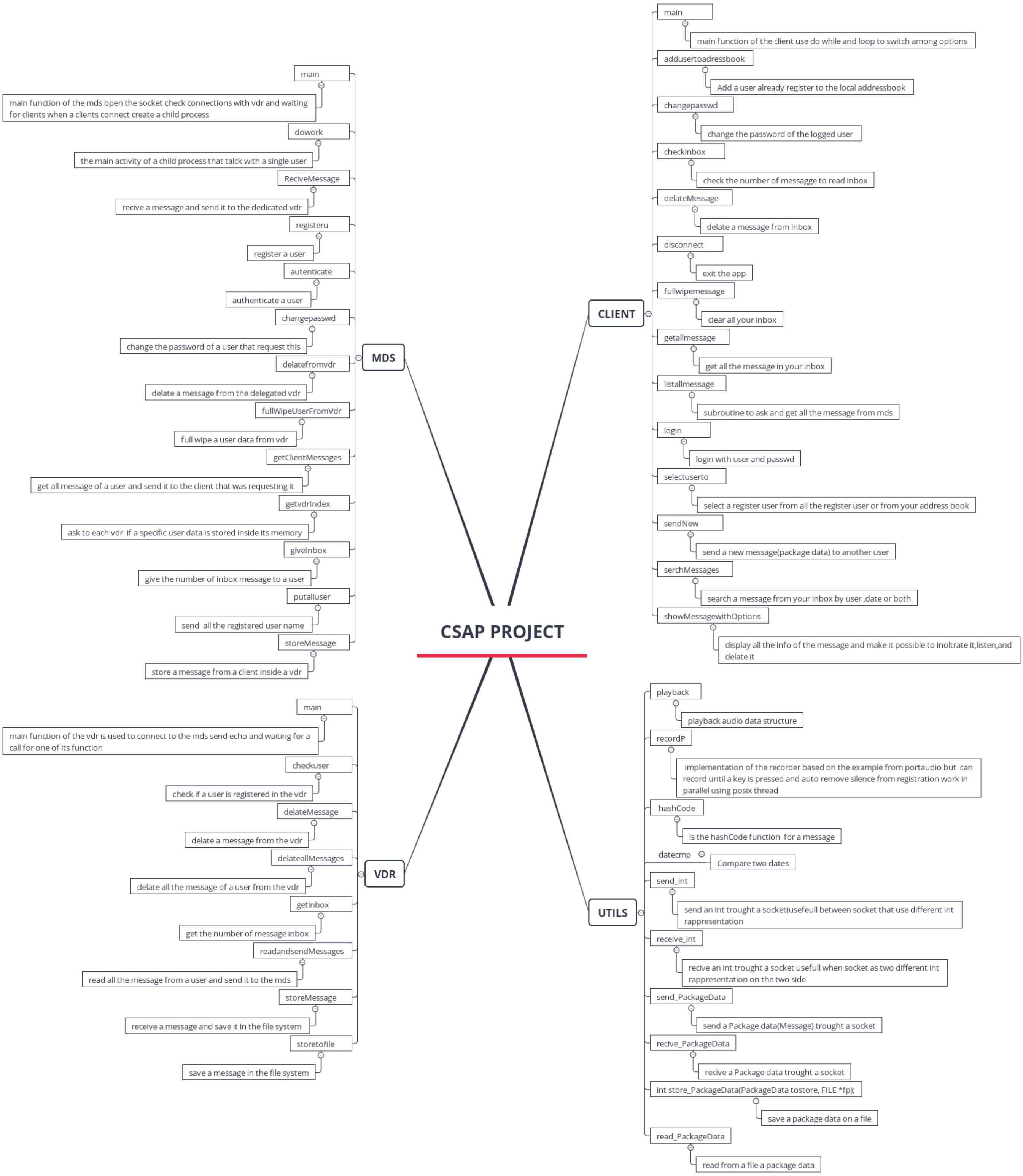
you already know

Release notable problems:

~~lib-portaudio work using alsa on linux and sometime i found problems in getting audio by the microphone off my laptop (Dell G3), anyway i have also problems with other app like zoom or google meet~~ (turns out that was *Intel* Sunrise Point-LP HD *Audio* devicd and is *Linux* driver snd_hda_intel related problem)

maybe recording is a little too verbose and config files too little

Project structure(each function and is use) :


CSAP PROJECT
MDS
main
main function of the mds open the socket check connections with vdr and waiting for clients when a clients connect create a child process
dowork
the main activity of a child process that talck with a single user
ReciveMessage
recive a message and send it to the dedicated vdr
registeru
register a user
autenticate
authenticate a user
changepasswd
change the password of a user that request this
delatefromvdr
delate a message from the delegated vdr
fullWipeUserFromVdr
full wipe a user data from vdr
getClientMessages
get all message of a user and send it to the client that was requesting it
getvdrIndex
ask to each vdr if a specific user data is stored inside its memory
giveInbox
give the number of inbox message to a user
putalluser
send all the registered user name
storeMessage
store a message from a client inside a vdr
CLIENT
main
main function of the client use do while and loop to switch among options
addusertoadressbook
Add a user already register to the local addressbook
changepasswd
change the password of the logged user
checkinbox
check the number of messagge to read inbox
delateMessage
delate a message from inbox
disconnect
exit the app
fullwipemessage
clear all your inbox
getallmessage
get all the message in your inbox
listallmessage
subroutine to ask and get all the message from mds
login
login with user and passwd
selectuserto
select a register user from all the register user or from your address book
sendNew
send a new message(package data) to another user
serchMessages
search a message from your inbox by user ,date or both
showMessagewithOptions
display all the info of the message and make it possible to inoltrate it,listen,and delate it
VDR
main
main function of the vdr is used to connect to the mds send echo and waiting for a call for one of its function
checkuser
check if a user is registered in the vdr
delateMessage
delate a message from the vdr
delateallMessages
delate all the message of a user from the vdr
getinbox
get the number of message inbox
readandsendMessages
read all the message from a user and send it to the mds
storeMessage
receive a message and save it in the file system
storetofile
save a message in the file system
UTILS
playback
playback audio data structure
recordP
implementation of the recorder based on the example from portaudio but can record until a key is pressed and auto remove silence from registration work in parallel using posix thread
hashCode
is the hashCode function for a message
datecmp
Compare two dates
send_int
send an int trought a socket(usefeull between socket that use different int rappresentation
receive_int
recive an int trought a socket usefull when socket as two different int rappresentation on the two side
send_PackageData
send a Package data(Message) trought a socket
recive_PackageData
recive a Package data trought a socket
int store_PackageData(PackageData tostore, FILE *fp);
save a package data on a file
read_PackageData
read from a file a package data

Some of the project requirements

Metadata Server (MDS)

● Handles initial connections with clients, maintains a “database” (even implemented as a file) containinginformation regarding users, messages, mailbox status, free space, etc

Upon startup, or after a crash, the MDS contacts all known VDR processes, which could be running on thesame or on another server, to get/check information about the status.

– Both the MDS and VDR keep copy of their own metadata.

The information for socket connections is contained in a configuration file,read upon startup by all processes

After initialization, the MDS waits for requests from clients

ONE or more Voice Data Repositories (VDR)

● Store audio files, keeping a copy of metadata (username,timestamp, ..) plus an hash for security.

● Send/Receive/Delete audio files upon request from MDS (checking hashes before returning them to server)

Clients

● Connect to MDS to authenticate

● Check for new messages

Search for messages:

– From specific users, selecting them from the entire list of from a personal address book – Within a specific data range

● Retrieve and play messages on the local audio system

● Delete messages

● Record a voice message and send it to a user (selected as indicated in the above bullet)