ChatBot for an IRC Server

Requirements

- 10 gb of RAM if the LLM is going to be hosted locally.
- If LLM will be curled from some other host, the requirements are very minimal.
- libcurl4 C library.
- libcjson C library.

Features

- Uses Ollama to generate responses (Currently token size is set to 30, it can be changed in message_compilator.h).
- Uses UNIX internals for smooth operation
- Monitors all defined channels at once.
- Logs channel messages and generated responses.
- Admin channel and admin commands.
- Ability to mute users (only allowed for admin user).
- Ability to shutdown the program, without leaving the IRC server (only allowed for admin user).
- Unique topic arguments for LLM response generation (Only allowed for admin).
- Ability to mute channels. (Only allowed for admin).

Missing Features

- Chat history for LLM is not stored.

Makefile

- Makefile is used to install required libraries with with Ollama if it is chosen to do so.
- Makefile has several functions. make install-libraries installs required libraries for compilation without ollama. make install-libraries-and-llama installs required libraries and ollama with llama 3.2 model. Please note that both of these functions should be executed with sudo permissions.
- After executing one of the previously mentioned functions, user should run make. After this command executes the user will have an executable file called client.

Program Workflow

When the program is launched it first checks whether all required configuration files are present. After file validation the program tries to connect to a specified server through socket connection. If connection fails or is disrupted, at any time of the program execution, the program will attempt to reconnect to it 5

times with each time increased waiting timer. After establishing connection the program loads up required config files, creates semaphores, allocates shared memory, creates pipes and signal handling. After creating UNIX internals the program starts to fork. Each defined server in the config file channels.cfg gets its own "server listener" process, it handles writing to the joined channel. One "server reader" process is created which reads the socket input from the server and sends out the read buffer to "channel listener" processes. One process for response generation is created (uses OLLAMA), this process receives data from "channel listener" processes and returns generated LLM response. One process is created for admin channel, it handles defined admin commands, channel specifications are stored in admin.cfg file. Lastly one process for logging is created, it stores received data from the server and sent data from the chat bot in a file which can be found inside logs/chat.log. If a signal for shutting down is received, the program kills all of the forked processes, frees allocated resources, closes file descryptors and shuts down.

Custom commands

These commands can be executed in the admin channel.

- ignore user0000. Up to 10 users if this limit is exceeded the chat bot will send a notification. Example ignore kaza5555. Currently user name is expected to not be longer than 9 symbols.
- poweroff - send a command to shutdown the bot without leaving the irc channel.
- donotchat #Unix - adds a channel to muted channel list, where the bot will not interact with anyone. Up to 32 channels, channel name is expected to be no longer than 63 symbols.
- topic number. Example topic 1. Sets a topic that LLM should reference its responses on. Currently only two topics are available. topic 1 = "Topic:Unix", topic 2 = "Topic:Cooking", topic 0 = "No specified topic". If the number is exceeded the bot will allert you or in a case of double digit the first number will be used and if it does not exceed the limitations appropriate topic will be selected.
- More will be added...

UNIX Internals

Sockets

- Used to connect to a specified server.

Forking

- Chat Bot runs several processes at once.
- One process is used for monitoring all the socket input sent from the server.

- Several processes (up to 32, depends on how many are defined in the config file) are used for interacting with IRC channels.
- One process is used for generating responses using a LLM.
- One process is used to monitor an admin channel and identify specified commands, and execute on them.
- One process is used to log all the data.

Semaphores

- Used for monitoring when log file is ready for writing and no other process is currently writing to shared memory.
- Used for allowing to generate only one message at a time using a LLM. If several requests are sent, other requests wait for a semaphore to open.
- Used for writting only one message at the time to the server. If several processes have something to write to the server, this semaphore takes care that only one message will be written at that time.

Pipes

- Every channel process uses pipes. They receive data from server reader process which identifies by from which channel it was received.
- Response generator process use pipes. It receives data from channel monitor process and then generates a response regarding the prompt given, and sends it back to the channel monitor process to send to the users.
- Admin channel process also uses pipes. It receives data from server reader process with user specified commands and then executes them.

Shared Memory

- Shared memory is used in several cases. One of them is to monitor if a socket is still alive. If at some point socket breaks bool flag is changed to false. This way the program knows whether this was an intended disconnect or a forced socket disconnection.
- Another use case is to store muted users. Admin channel collects muted users and then later on server reader process can ignore messages sent from those users.
- Shared memory is also used for logging messages. Channel monitor processes write into a shared memory buffer, once that buffer is written semaphore is turned on and is not turned off until the message is written into the file.
- Shared memory also stores muted channels. An admin can create a new rule to ignore all input in a certain channel by sending a command in admin channel. All chatting will be ignored in specified channel.
- Shared memory is also used for storing specific topics for LLM response generation and changing the selected topic. Executing specific command in admin channel changes the way LLM generates its responses to specific questions.

Signals

- Signals are used to gracefully close the program when a CTRL + C or a SIGINT signal is received. This approach is used twice, once to stop currently running processes and gracefully exit the program, another time is to gracefully shutdown the program when it is trying to reconnect a socket.
- Signals are also used to wake up logger process when new data in shared memory buffer is available.

File Structure

Config

- Stores all config files.
- admin.cfg is a configuration file for creating an admin channel. First line specifies channel name, second line specifies channel password.
- channels.cfg is a configuration channel for joining channels. Channel for joining should be always written in a new line. Program currently supports up to 32 channels.

Include

- Stores C header files and additional information.
- connection_handler.h stores server ip address, server port and user name for the bot to use.
- message_compilator.h stores OLLAMA model that will be using in the program and a url to generate responses from. It also stores the length of allowed token for OLLAMA generation, currently set to 30 to speed up response generation process.

Logs

- Stores chat.log file. All logging is placed there.

Responses

- Used for generating OLLAMA responses.

Src

- C source files.