

---

# *Apiza: The API Assistant Study*

---

## **Description**

In this study, you will be asked to complete a series of tasks involving the use of the libssh API. To assist you with these tasks, you will be connected with our experimental virtual assistant, **Apiza**.

As you work through these tasks, you will have to make use functions and constants from the libssh API. When you have any questions about the API, such as determining an appropriate function or learning about function parameters, we ask that you direct your question to **Apiza**.

If you have other general programming questions, feel free to consult internet resources (Google, StackOverflow, etc). However, avoid mentioning libssh or any of its components in your queries, as all of those questions should be directed towards Apiza.

The tasks should be completed in order, as each one builds off of the previous ones. If you run into any issues, please inform the study supervisor. You will be compensated for your participation, so take your time and do your best. It is alright if you do not complete all of the tasks.

## **Set-up**

If you have brought your own machine, make sure VirtualBox is installed. Import the provided disk image and start the virtual machine.

Login to the “Participant” account using the password “password”. Open Firefox. All internet browsing in this study should be done through Firefox. Navigate to Slack. Login to the “Apiza” workspace. Click on the user “Apiza” to begin a chat session with the virtual assistant.

Using whichever text editor you prefer, complete the tasks described on the next page. The only file you will need to modify is ~/Documents/Tasks/ssh\_tasks.c

## **Tasks**

1. Compile the `ssh_tasks.c` file using CMake (Open the terminal, navigate to the folder `~/Documents/Tasks/`, and run `make`). Run the compiled executable.
2. Note that the `libssh` library is included in the beginning of the program:

```
#define LIBSSH_STATIC 1
#include <libssh/libssh.h>
```

In the `main` method, use the `libssh` API to create a new `ssh_session`. Ensure that the session is successfully created.

3. In the `main` method, finish setting up the session.
  - Connect to the `localhost`.
  - Authenticate the server (check that it is in the known host file) and authenticate the user to the server.
  - Disconnect from the server and free the session.

If there are any errors, immediately report them and `exit` the program.

4. Write the function `show_remote_user(ssh_session session)`.
  - Create and open a new `ssh_channel` in the `ssh_session`.
  - Execute the command `"who"` on the open channel.
  - Read in response from the channel and print it to `stdout`.
  - Shut down the channel by sending the eof and closing and freeing the channel.

If there are any errors, immediately return the error code. Otherwise, return the default status code.

5. Write the function `sftp_operations(ssh_session session)`.
  - Create and initialize a new `sftp_session` on the connected `ssh_session`.
  - Create a new directory `"Dir1"`. Return any errors, but do not return if the directory already exists.
  - Create a new file in `Dir1` called `"File1"`.
  - Write the string `"Hello, world!"` to the file.
  - Close the file, and free the `sftp` session.

If there are any errors, immediately return the error code. Otherwise, return the default status code.