Program Project

1 Task

Build a storage-as-a-service application with Python. The application works in the client / server model, whereby the client logs on to the server and can perform various operations. All operations and their codes are described in Table 1. The codes of the operations are required for the exchange of information over the network.
The environment is largely controlled on the client side, i.e., the client sends the Op-Codes for the desired activities to the server, the server reacts on the basis of these Op-Codes and transmits the corresponding data or other Op-Codes. If the client sends unknown commands, the server replies with *RST*.
The server offers user management so that users can only see their own files. The client logs on to the server using a username / password. If the registration has not been carried out correctly, no other Op-Codes except *LOGIN* and *FIN* are possible.
For each username, it is also determined whether a user has administration rights or not. Administrators have extended rights and can perform additional operations.
A logged in user can only see his own files and only upload and download these. Administrators can see all directories and data belonging to the file hosting server.
The server also only works on the command line; no graphical user interface is required. It logs all relevant information in order to be able to understand activities carried out. The level of detail of the log is not fixed but can be determined independently. The data (user data, user files, log files) should be stored persistently in a suitable environment (file system, database, etc.).

Table 1: Operations in the area

Op-Code	**Meaning**	**root-commando**
ACK	Confirmation of correctness	-
RST	Verification failed	-
SI	Request more information	-
LOGIN	Send login data	-
LST	List the user's files on the server	-
UP	Upload the file to the user's directory	-
DL	Download the file to the user's directory	-
ST	Query statistics	x
NU	Create new user	x
DU	Delete user	x
FIN	End communication	-

2 Subtasks

The following sections describe some areas of the application in more detail.

2.1 Login Prozess

The client uses the Op-Ccode *LOGIN* to report the request that it would like to log on to the server. If the server is able to do this, it also replies with *LOGIN*. The client must then transmit the data in the form *USER*:*username*:*PASS*:*password*,
e.g., USER:*admin*:*PASS*:*password*.
If the registration is correct, the server confirms the correctness with the Op-Code *ACK* and waits for the next operation, otherwise an *RST* is sent so that the data must be sent again.
The login data can be given as a parameter when calling the client,
e.g.

```
pyton3  client.py  user=user1  pw=123
```

or otherwise queried on the client.

2.2 Data exchange

The data exchange takes place in the following form:
As soon as the client has announced the upload of a file via Op-Code *UP*, the server uses Op-Code *SI* to send the request to the client to transmit further details about the file. This includes the file name and the file size in bytes.
The client transmits the data in the form
File name — file size, e.g., testfile.txt — 133 for a file with the name *testfile.txt* and a total size of 133 bytes.

Table 2: users and passwords

User	**Password**	**Admin**
admin	123456	No
user1	user1	No
user2	qwertz	Yes
root	password	Yes

If the client wants to download data, it transmits the information in the form *DL:filename*, e.g. *DL:testfile.txt*, the server transmits the file and then waits for new commands. If no data is transmitted within the defined timeout period (this corresponds to 15 seconds), the connection request is rejected on both sides and the client can send new Op-Codes.

2.3 User management

If the client wants to create a new user via *NU*, the server sends its readiness via *NU*.
The client now transmits the data in an adapted form to the known format. An ADMIN parameter is added, which grants the user extended rights. If the parameter ADMIN: 1 is used, the user has extended rights, with ADMIN: 0 he has no extended rights.
The correct form is therefore *USER:username:PASS:password:ADMIN:1* to the server.
This saves the data in its persistent database so that a login is possible later.
The user with passwords listed in Table 2 must already be stored on the system by default.
The server is informed by means of the Op-Code *DU* that a user account is to be deleted. The syntax *DU:username* is used for this, e.g. *DU:user1*. If the account exists, the account is deleted and confirmed with *ACK*, otherwise an error is reported via *RST*.
Whether the associated files are also deleted when a user is deleted is not defined and can be freely implemented.

2.4 Status display

If a user logs in with the appropriate authorization, he can use the Op-Code *ST* to query special statistical information about the system.
The information about statistical data can be chosen almost freely, the implementation is also arbitrary. The information should be presented in text form; the client must take care of the correct formatting of the data supplied.

However, at least the following information should be available:

- Number of data per user
- Amount of data per user
- Current runtime oft he server
- Number of incorrect logins per user

3 Framework

It is essential for the assessment that the client and the server work together flawlessly not only with their own implementation, but also with other students implementations. For this to be successful, the following framework conditions must be observed during implementation:

- Communication between client and server takes place via TCP
- The used port is transferred to both the client and the server via parameters at startup, so that the port can be selected flexibly
- Use of Op-Codes according to Table 1
- The login data according to table 2 are available on the server
- A Server can manage several clients in parallel

In addition, the other requirements that must be complied with for a clearer structure also apply:

- The entire storage of data takes place locally in a directory with corresponding subdirectories. For example, such a structure is possible for a directory

```
$ tree projekt
projekt
 |-- logs
 |-- userdata
  |--user1
  |--user2
  |--user3
 |-- management
```

- There is no need to secure the data by accessing the data from 'outside'. The access restriction should only apply within the python program
- Avoid using a large number of external libraries (i.e. libraries to be installed separately)

4 Course of the program

One possible sequence oft he program is as follows:

1. Server is waiting for connection
2. Client establishes connection and registers *LOGIN*
3. Server asks for login data
4. Client transfers the data in form *USER:username:PASS:Passwort*
    - If the data are incorrect, a reset is carried out (Op-Code *RST*)
    - If the data is correct, the user can query his data (via *LST*), access his files (via *DL*) or add new ones (via *UP*)
5. When listing, the server shows all data (possibly including existing metadata) that are assigned to the user
6. During the download, the client transmits the file name in the formal syntax and then waits for the data to be made available
7. During the upload, the server asks for further details via *SI*
8. The client transmits the file name and size in the formal syntax
9. The client send the relevant information (e.g. name) tot he server and waits fort he data
10. The server sends the data