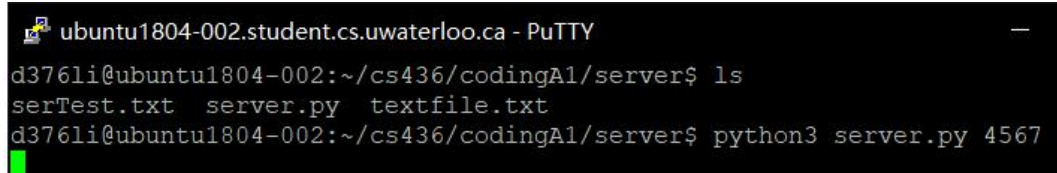


Instruction

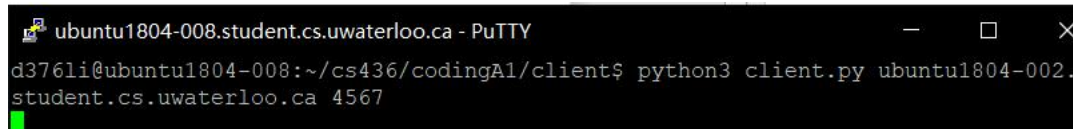
The client-end program is implemented in client.py, and the server-end program is implemented in server.py. To run server.py, execute `python3 server.py <n_port>` in the terminal. Here we use `<n_port> = 4567` and execute it in `ubuntu1804-002.student.cs.uwaterloo.ca`, which is shown in Fig.1.



```
ubuntu1804-002.student.cs.uwaterloo.ca - PuTTY
d3761i@ubuntu1804-002:~/cs436/codingA1/server$ ls
serTest.txt  server.py  textfile.txt
d3761i@ubuntu1804-002:~/cs436/codingA1/server$ python3 server.py 4567
```

Fig.1 Execution of server.py

Then, we run `python3 client.py <server_address> <n_port>` to turn on the client end, which is shown in Fig.2.

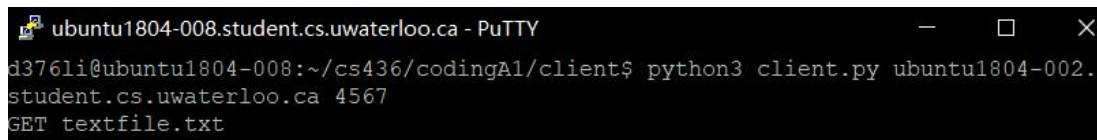


```
ubuntu1804-008.student.cs.uwaterloo.ca - PuTTY
d3761i@ubuntu1804-008:~/cs436/codingA1/client$ python3 client.py ubuntu1804-002.
student.cs.uwaterloo.ca 4567
```

Fig.2 Execution of client.py

Here we can input command to either GET, PUT files or EXIT.

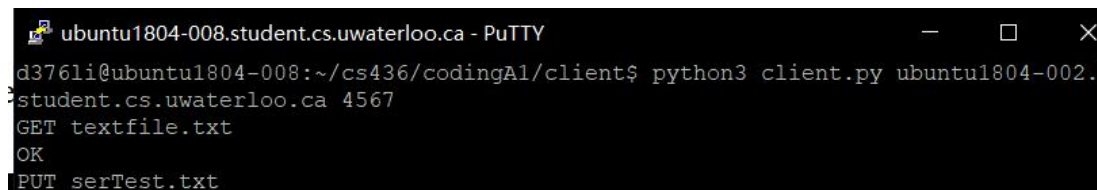
For GET: Input `GET <file_name>` to download files from the server as demonstrated in Fig.3.



```
ubuntu1804-008.student.cs.uwaterloo.ca - PuTTY
d3761i@ubuntu1804-008:~/cs436/codingA1/client$ python3 client.py ubuntu1804-002.
student.cs.uwaterloo.ca 4567
GET textfile.txt
```

Fig.3 GET command

For PUT: Input `PUT <file_name>` to upload files from the server as demonstrated in Fig.4.



```
ubuntu1804-008.student.cs.uwaterloo.ca - PuTTY
d3761i@ubuntu1804-008:~/cs436/codingA1/client$ python3 client.py ubuntu1804-002.
student.cs.uwaterloo.ca 4567
GET textfile.txt
OK
PUT serTest.txt
```

Fig.4 PUT command

For EXIT: Input `EXIT` to terminate the client end, which is shown in Fig.5.

```
ubuntu1804-008.student.cs.uwaterloo.ca - PuTTY
d376li@ubuntu1804-008:~/cs436/codingA1/client$ python3 client.py ubuntu1804-002
student.cs.uwaterloo.ca 4567
GET textfile.txt
OK
PUT serTest.txt
OK
EXIT
d376li@ubuntu1804-008:~/cs436/codingA1/client$
```

Fig.5 EXIT command

During the whole process, the server-end program is always-on as Fig.6.

```
ubuntu1804-002.student.cs.uwaterloo.ca - PuTTY
d376li@ubuntu1804-002:~/cs436/codingA1/server$ python3 server.py 4567
```

Fig.6 Server-end program state

Finally, we can see the client successfully downloaded the file “serTest.txt”, and the server successfully received the file “textfile.txt”, whose statuses are shown in Fig.7 and Fig.8.

```
ubuntu1804-008.student.cs.uwaterloo.ca - PuTTY
d376li@ubuntu1804-008:~/cs436/codingA1/client$ python3 client.py ubuntu1804-002.
student.cs.uwaterloo.ca 4567
GET textfile.txt
OK
PUT serTest.txt
OK
EXIT
d376li@ubuntu1804-008:~/cs436/codingA1/client$ ls
client.py  serTest.txt  textfile.txt
d376li@ubuntu1804-008:~/cs436/codingA1/client$
```

Fig.7 Files in client’s folder

```
ubuntu1804-002.student.cs.uwaterloo.ca - PuTTY
d376li@ubuntu1804-002:~/cs436/codingA1/server$ ls
serTest.txt  server.py  textfile.txt
d376li@ubuntu1804-002:~/cs436/codingA1/server$
```

Fig.8 Files in server’s folder

Test machine

As claimed in the instruction section, the server.py is run on ubuntu1804-002.student.cs.uwaterloo.ca, and client.py is run on ubuntu1804-008.student.cs.uwaterloo.ca.

Reference

Python Socket Mannual. (n.d.). Retrieved March 02, 2021, from
<https://manpages.debian.org/buster/manpages-dev/recv.2.en.html>