Grass sample tests

In test output, bold texts are dynamic and may change in different runs. However, normal texts are deterministic and require an **exact match**.

Test 1: ping, date

Base directory:

```
Empty
```

Client input:

```
login u1
pass p1
date
ping 8.8.8.8
ping epfl.ch
ping nonexistingaddress
date
```

Clint output/Server output:

```
Tue Nov 5 16:00:00 CEST 2019
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=17.1 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 17.105/17.105/17.105/0.000 ms

PING epfl.ch (128.178.222.108) 56(84) bytes of data.
64 bytes from www-origin.epfl.ch (128.178.222.108): icmp_seq=1 ttl=128
time=11.1 ms

--- epfl.ch ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 11.129/11.129/11.129/0.000 ms
ping: nonexistingaddress: Name or service not known
Tue Nov 5 16:00:15 CEST 2019
```

Test 2: Multiple clients

- ‘w’ prints users in alphabetical order.

- ‘exit’ can only be the last command and it closes the client.
- We will not use interrupts or exit command in the middle of your program.

Base directory:

```
Empty
```

Client 1 input:

```
login u1
pass p1
whoami
w
```

Client 1 output:

```
u1
u1
```

Client 2 input:

```
login u2
pass p2
whoami
w
exit
```

Client 2 output:

```
u2
u1 u2
```

Client 3 input:

```
login u3
pass p3
whoami
w
```

Client 3 output:

```
u3
u1 u3
```

Server output:

```
u1
u1
u2
u1 u2
u3
u1 u3
```

Test 3: Directories

Base directory:

Empty

Client input:

```
login u1
pass p1
ls
cd dir1
mkdir dir1
cd dir1
ls
mkdir dir2
ls
rm dir2
ls
cd ..
ls
```

Clint output/Server output:

```
total 0
cd: dir1: No such file or directory
total 0
total 4
drwxr-xr-x 2 root root 4096 Nov 5 16:00 dir2
total 0
total 4
drwxr-xr-x 2 root root 4096 Nov 5 16:00 dir1
```

Test 4: Transfer

- ‘get’ saves the file in the client’s executable directory.

Base directory:

```
total 4
-rw-r--r-- 1 root root  136 Nov 5 14:00 server_file.txt
```

Client directory:

```
-rwxr-xr-x 1 root root 621 Nov 5 15:00 client.exe
-rw-r--r-- 1 root root  74 Nov 5 15:00 client_file.txt
```

Client input:

```
login u1
pass p1
ls
get server_file.txt
ls
put client_file.txt 74
ls
```

Clint output/Server output:

```
total 4
-rw-r--r-- 1 root root  136 Nov 5 14:00 server_file.txt
total 4
-rw-r--r-- 1 root root  136 Nov 5 14:00 server_file.txt
total 8
-rw-r--r-- 1 root root  74 Nov 5 16:00 client_file.txt
-rw-r--r-- 1 root root 136 Nov 5 14:00 server_file.txt
```

Test 5: Authentication

- If a user tries to perform a privileged command without performing an authentication, then he/she would receive an access denied error. Same goes for performing pass without asking login first.
- A failed authentication should respond with “Authentication failed.”

Base directory:

```
Empty
```

Client input:

```
ls
login u1
ls
pass p1
login u1
pass wrong-pass
login u1
pass p1
whoami
ls
login u2
pass p2
whoami
logout
whoami
ls
```

Clint output/Server output:

```
Error: access denied.
Error: access denied.
Error: access denied.
Authentication failed.
u1
total 0
u2
Error: access denied.
Error: access denied.
```

Test 6: Errors

Base directory:

```
total 4
-rw-r--r-- 1 root root  136 Nov 5 16:00 server_file.txt
```

Client input:

```
login u1
pass p1
ls
cd ../../../mnt
ls
Put loooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooong 200
rm ../..
mkdir looooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooong
```

Clint output/Server output:

```
total 4
-rw-r--r-- 1 root root  136 Nov 5 16:00 server_file.txt
Error: access denied.
total 4
-rw-r--r-- 1 root root  136 Nov 5 16:00 server_file.txt
Error: the path is too long.
Error: access denied.
Error: the path is too long.
```

Test 7: Grep

- ‘grep’ outputs file addresses in alphabetical order.

Base directory:

```
dir1/F1.txt: “This is file 1.”
A-F1.txt: “This is a file.”
F1.txt: “This is file 1.”
F2.txt: “This is file 2.”
F3.txt: “Nothing here.”
Sample.txt: “Sample.”
```

Client input:

```
login u1
pass p1
```

```
grep file
```

Clint output/Server output:

```
A-F1.txt
dir1/F1.txt
F1.txt
F2.txt
```

Test 8: Grep 2

Base directory:

```
F1.txt: “some random text”
F2.txt: “some other random text”
dir1/F1.txt: “random.guy@epfffl.ch”
dir1/dir2/F1.txt: “random_guy@epfl.com”
dir1/dir2/F2.txt: “random.guy@epfl.ch”
dir1/F2.txt: “randomguy@epfl.ch”
```

Client input:

```
login u1
pass p1
grep @epfl.ch
```

Clint output/Server output:

```
dir1/dir2/F1.txt
dir1/dir2/F2.txt
dir1/F2.txt
```

Test 9: Performance

Base directory:

```
File0001.txt: “This is file 0001.”
File0002.txt: “This is file 0002.”
File0003.txt: “This is file 0003.”
....
File9999.txt: “This is file 9999.”
```

Client input:

```
login u1
pass p1
grep 7
```

Clint output/Server output:

```
File0007.txt
File0017.txt
...
File0070.txt
...
File9997.txt
```