Name: Tan Han Nguyen

NetID: TXN200004

Week 12 Lab 1

0. Login

Pre-authentication banner message from server:

| University of Texas at Dallas

| Department of Computer Science

|

| Use of UTD Information Systems is subject to

| the UTD Information Security and Acceptable Use Policy.

|

| Pursuant to Texas Administrative Code 202:

| (1) Unauthorized use is prohibited;

| (2) Usage may be subject to security testing and monitoring;

| (3) Misuse is subject to criminal prosecution; and

| (4) No expectation of privacy except as otherwise provided by applicable

| privacy laws.

|

| ATTENTION: utdnetid != utdnetid@utdallas.edu (UTD != Google!)

|

| \*\*\*\*\* This system will require a connection to the GlobalProtect VPN startin

> g

| on the following dates:

|

| cslinux1.utdallas.edu - June 15, 2020

| cslinux2.utdallas.edu - June 22, 2020

|

| \*\*\*\*\* GlobalProtect VPN Instructions: https://www.utdallas.edu/oit/howto/vpn

> /

|

End of banner message from server

Keyboard-interactive authentication prompts from server:

| Password:

End of keyboard-interactive prompts from server

┌──────────────────────────────────────────────────────────────────────┐

│ • MobaXterm Personal Edition v24.2 • │

│ (SSH client, X server and network tools) │

│ │

│ ⮞ SSH session to txn200004@cslinux2.utdallas.edu │

│ • Direct SSH : ✓ │

│ • SSH compression : ✓ │

│ • SSH-browser : ✓ │

│ • X11-forwarding : ✓ (remote display is forwarded through SSH) │

│ │

│ ⮞ For more info, ctrl+click on help or visit our website. │

└──────────────────────────────────────────────────────────────────────┘

Last login: Sat Nov 2 21:30:30 2024 from 10.50.242.44

\*\*\*---\*\*\*---\*\*\*---\*\*\*---\*\*\*---\*\*\*

csgrads1.utdallas.edu - CentOS Linux 7.9

--All CS Graduate Students should use csgrads1--

cs1.utdallas.edu - CentOS Linux 7.9

cs2.utdallas.edu - CentOS Linux 7.9

\*\*\*---\*\*\*---\*\*\*---\*\*\*---\*\*\*---\*\*\*

This system is for use by CS students who need a general purpose Linux system

to complete homework assignments. Computationally or resource intensive

simulations will be throttled automatically.

Thank you,

CS Lab Manager

cs-labs@utdallas.edu

/scratch disk space can be used for temporary files.

All files within /scratch will be erased on a regular basis (Sunday 0300).

{cslinux2:~} mkdir week12Lab1; cd week12Lab1

1. Update timeServer0.c and timeClient0.c

{cslinux2:~/week12Lab1} vim timeServer0.c

{cslinux2:~/week12Lab1} vim timeClient0.c

List timeServer0.c

{cslinux2:~/week12Lab1} cat timeServer0.c

// server0.c - listening port# 12345

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <time.h>

#include <signal.h>

#define PORT 22719

//return 1 if there's a previous server running. 0 otherwise

int check\_running\_server (char \* prog\_name, int curr\_pid){

int return\_val = 0;

//export previous running server to a temp file

char command[100];

snprintf(command, sizeof(command), "ps -f | grep -v %d | grep -v grep | grep \' %s\' > .server\_check.temp", curr\_pid, prog\_name);

system(command);

//open temp file

FILE \*fp = fopen(".server\_check.temp", "r");

if (fp == NULL) {

perror("Error opening file");

exit(1);

}

//Extracting PID from .server\_check.temp

char line[1024];

while(fgets(line, sizeof(line), fp) != NULL) {

int pid;

sscanf(line, "%\*s %d", &pid);

printf("\n\*\*timeServer: Found running server PID: %d\n", pid);

printf("\n\*\*timeServer: Killing running server PID: %d...\n", pid);

kill(pid, SIGKILL);

return\_val = 1;

}

fclose(fp);

system("rm -f .server\_check.temp"); //delete temp file

return return\_val;

}

int main(int argc, char \*argv[])

{

int curr\_pid = getpid();

if (check\_running\_server(argv[0], curr\_pid) == 1) {

printf("\n\*\*timeServer: Waiting for port to free before connecting...\n");

//Wait until the port is free using a temp file

int is\_not\_empty = 1;

while(is\_not\_empty) {

char command[100];

snprintf(command, sizeof(command), "netstat -aont | grep \"%d\" > .check\_port.temp", PORT);

system(command);

FILE \*fp = fopen(".check\_port.temp", "r");

if (fp == NULL) {

perror("File open failed");

return -1; // Error in accessing the file

}

is\_not\_empty = (fgetc(fp) != EOF); // Check if first character is EOF

fclose(fp);

if(is\_not\_empty){

printf("\nPort %d still in use...\n", PORT);

sleep(5);

}

}

system("rm -f .check\_port.temp"); //delete temp file

printf("\n\*\*timeServer: Done waiting! Port %d is freed and ready to bind\n", PORT);

//sleep(60);

}//check and kill previous server

int listenfd = 0, connfd = 0;

struct sockaddr\_in serv\_addr;

char sendBuff[1025];

time\_t ticks;

listenfd = socket(AF\_INET, SOCK\_STREAM, 0);

memset(&serv\_addr, 0, sizeof(serv\_addr));

memset(sendBuff, 0, sizeof(sendBuff));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

serv\_addr.sin\_port = htons(PORT);

if (bind(listenfd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)) == -1) {

perror("bind failed\n");

return -1;

}

if(listen(listenfd, 10) == -1) {

perror("listen\n");

return -1;

}

printf("\n\*\*timeServer: Server is up and listening through Port %d...\n", PORT);

system("ps");

char command[100];

snprintf(command, sizeof(command), "netstat -aont | grep \"%d\"", PORT);

system(command);

while(1)

{

connfd = accept(listenfd, (struct sockaddr\*)NULL, NULL);

if(connfd == -1){

perror("accept");

return -1;

}

printf("\n\*\*timeServer: A client connected \n");

ticks = time(NULL);

snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n", ctime(&ticks));

printf("\n\*\*timeServer: Writing to client...\n");

if(write(connfd, sendBuff, strlen(sendBuff)) == -1) {

perror("write\n");

return -1;

}

close(connfd);

printf("\n\*\*timeServer: Client disconnected \n\n");

sleep(1);

}

}

List timeClient0.c

{cslinux2:~/week12Lab1} cat timeClient0.c

// client0.c connects to server listening port# 12345

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <arpa/inet.h>

#define PORT 22719

int main(int argc, char \*argv[])

{

char ip[] = "127.0.0.1"; // default IP of the server

argv[1] = ip;

int sockfd = 0, n = 0;

char recvBuff[1024];

struct sockaddr\_in serv\_addr;

memset(recvBuff, 0,sizeof(recvBuff));

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

printf("\n Error : Could not create socket \n");

return 1;

}

memset(&serv\_addr, 0, sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

if(inet\_pton(AF\_INET, argv[1], &serv\_addr.sin\_addr)<=0)

{

printf("\n inet\_pton error occured\n");

return 1;

}

serv\_addr.sin\_port = htons(PORT);

system(" date; hostname; whoami ");

system(" netstat -aont | grep \":2271[0-9]\"");

printf("\n timeClient: connecting to %s Port# = %d \n", ip, PORT);

if( connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0)

{

printf("\n Error : Connect Failed \n");

return 1;

}

printf("\n timeClient: connected to timeServer. \n");

system("ps");

system(" netstat -aont | grep \":2271[0-9]\"");

printf("\n\n");

printf("\n timeClient: reading from timeServer. \n");

while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)

{

recvBuff[n] = 0;

if(fputs(recvBuff, stdout) == EOF)

{

printf("\n Error : Fputs error\n");

}

}

if(n < 0)

{

printf("\n Read error \n");

exit(0);

}

printf("\n timeClient: now terminated. \n");

return 0;

}

Compile and run timeServer0 and timeClient0

{cslinux2:~/week12Lab1} gcc timeServer0.c -o timeServer0

{cslinux2:~/week12Lab1} gcc timeClient0.c -o timeClient0

2. Use ps and netstat to check the state of server and client

{cslinux2:~/week12Lab1} ps

PID TTY TIME CMD

45770 pts/0 00:00:00 bash

46192 pts/0 00:00:00 ps

{cslinux2:~/week12Lab1} netstat -aont | grep 2271[0-9]

Before running ./timeServer0 there’s no process or port connection (all ports are free)

{cslinux2:~/week12Lab1} ./timeServer0 &

[1] 46214

{cslinux2:~/week12Lab1}

\*\*timeServer: Server is up and listening through Port 22719...

PID TTY TIME CMD

45770 pts/0 00:00:01 bash

46214 pts/0 00:00:00 timeServer0

46221 pts/0 00:00:00 ps

tcp 0 0 0.0.0.0:22719 0.0.0.0:\* LISTEN off (0.00/0/0)

After running ./timeServer0 we can see the server process with PID 46214

Also the server is listening through port 22719 for TCP connection

./timeClient0

Sat Nov 2 21:39:56 CDT 2024

cslinux2.utdallas.edu

txn200004

tcp 0 0 0.0.0.0:22719 0.0.0.0:\* LISTEN off (0.00/0/0)

timeClient: connecting to 127.0.0.1 Port# = 22719

timeClient: connected to timeServer.

\*\*timeServer: A client connected

\*\*timeServer: Writing to client...

\*\*timeServer: Client disconnected

PID TTY TIME CMD

45770 pts/0 00:00:01 bash

46214 pts/0 00:00:00 timeServer0

46245 pts/0 00:00:00 timeClient0

46253 pts/0 00:00:00 ps

tcp 0 0 0.0.0.0:22719 0.0.0.0:\* LISTEN off (0.00/0/0)

tcp 0 1 127.0.0.1:22719 127.0.0.1:50268 FIN\_WAIT1 on (0.16/0/0)

tcp 27 0 127.0.0.1:50268 127.0.0.1:22719 CLOSE\_WAIT off (0.00/0/0)

timeClient: reading from timeServer.

Sat Nov 2 21:39:56 2024

timeClient: now terminated.

After running ./timeClient0 we can see that:  
1. Both process timeServer0 (46214) and timeClient0 (46245) are running

2. TCP connection established between the server and the client

3. The client successfully read the time from the server and then terminate

{cslinux2:~/week12Lab1} ps

PID TTY TIME CMD

45770 pts/0 00:00:01 bash

46214 pts/0 00:00:00 timeServer0

46283 pts/0 00:00:00 ps

{cslinux2:~/week12Lab1} netstat -aont | grep 2271[0-9]

tcp 0 0 0.0.0.0:22719 0.0.0.0:\* LISTEN off (0.00/0/0)

tcp 0 0 127.0.0.1:22719 127.0.0.1:50268 TIME\_WAIT timewait (34.96/0/0)

Run ps and netstate commands right after the client terminated we see:  
1. Client process is terminated (no PID)  
2. Server is still running (PID 46214)  
3. The connection has been terminated but the port is still under TIME\_WAIT until it’s completely free to use again.

\*\*Try to run a new timeServer0 process to test

{cslinux2:~/week12Lab1} ./timeServer0 &

[2] 46299

{cslinux2:~/week12Lab1}

\*\*timeServer: Found running server PID: 46214

\*\*timeServer: Killing running server PID: 46214...

\*\*timeServer: Waiting for port to free before connecting...

Port 22719 still in use...

Port 22719 still in use...

Port 22719 still in use...

Port 22719 still in use...

Port 22719 still in use...

Port 22719 still in use...

\*\*timeServer: Done waiting! Port 22719 is freed and ready to bind

\*\*timeServer: Server is up and listening through Port 22719...

PID TTY TIME CMD

45770 pts/0 00:00:01 bash

46299 pts/0 00:00:00 timeServer0

46356 pts/0 00:00:00 ps

tcp 0 0 0.0.0.0:22719 0.0.0.0:\* LISTEN off (0.00/0/0)

The new timeServer0 process checks for the previous running server (PID 46214), kill it, and wait until the port is freed then bind to the port and start listening again.

We can verify this by the ps command output: the old server PID 46214 is gone and replaced with a new server PID 46299.

\*try connecting client to the new server

./timeClient0

Sat Nov 2 21:41:11 CDT 2024

cslinux2.utdallas.edu

txn200004

tcp 0 0 0.0.0.0:22719 0.0.0.0:\* LISTEN off (0.00/0/0)

timeClient: connecting to 127.0.0.1 Port# = 22719

timeClient: connected to timeServer.

\*\*timeServer: A client connected

\*\*timeServer: Writing to client...

\*\*timeServer: Client disconnected

PID TTY TIME CMD

45770 pts/0 00:00:01 bash

46299 pts/0 00:00:00 timeServer0

46382 pts/0 00:00:00 timeClient0

46390 pts/0 00:00:00 ps

tcp 0 0 0.0.0.0:22719 0.0.0.0:\* LISTEN off (0.00/0/0)

tcp 0 0 127.0.0.1:22719 127.0.0.1:50274 FIN\_WAIT2 timewait (59.58/0/0)

tcp 27 0 127.0.0.1:50274 127.0.0.1:22719 CLOSE\_WAIT off (0.00/0/0)

timeClient: reading from timeServer.

Sat Nov 2 21:41:11 2024

timeClient: now terminated.

Successfully connect and read time from the new server process.

3. Kill the server before ending lab

{cslinux2:~/week12Lab1} ps

PID TTY TIME CMD

45770 pts/0 00:00:01 bash

46299 pts/0 00:00:00 timeServer0

46398 pts/0 00:00:00 ps

{cslinux2:~/week12Lab1} kill 46299

{cslinux2:~/week12Lab1} ps

PID TTY TIME CMD

45770 pts/0 00:00:01 bash

46423 pts/0 00:00:00 ps

[2]+ Terminated ./timeServer0 \*server process terminated

{cslinux2:~/week12Lab1} netstat -aont | grep 2271[0-9]

tcp 0 0 127.0.0.1:22719 127.0.0.1:50274 TIME\_WAIT timewait (21.75/0/0) \*waiting for port to free

{cslinux2:~/week12Lab1} netstat -aont | grep 2271[0-9]

\*port is freed here

End Lab and Exit

{cslinux2:~/week12Lab1} date

Sat Nov 2 21:42:25 CDT 2024

{cslinux2:~/week12Lab1} uname -a

Linux cslinux2.utdallas.edu 3.10.0-1160.119.1.el7.x86\_64 #1 SMP Tue Jun 4 14:43:51 UTC 2024 x86\_64 x86\_64 x86\_64 GNU/Linux

{cslinux2:~/week12Lab1} ls -l

total 96

-rwx--x--x 1 txn200004 se 8904 Nov 2 21:39 timeClient0

-rw------- 1 txn200004 se 1906 Nov 2 21:38 timeClient0.c

-rwx--x--x 1 txn200004 se 13736 Nov 2 21:39 timeServer0

-rw------- 1 txn200004 se 3955 Nov 2 21:37 timeServer0.c

{cslinux2:~/week12Lab1} exit

logout

───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────

Session stopped

- Press <Return> to exit tab

- Press R to restart session

- Press S to save terminal output to file