Name: Tan Han Nguyen

NetID: TXN200004

Week 12 Lab 2

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: Wed Nov 6 16:50:22 2024 from 10.50.240.241

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

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 week12Lab2; cd week12Lab2

1-4. Implement Server2.c Client2.c

{cslinux2:~/week12Lab2} vim Server2.c

{cslinux2:~/week12Lab2} vim Client2.c

List Server2.c

{cslinux2:~/week12Lab2} cat Server2.c

// Server2.c

#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>

//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\*\*SERVER\*\*: Found running server PID: %d\n", pid);

printf("\n\*\*SERVER\*\*: 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;

}

void handle\_sigalrm(int sig){

printf("\*\*SERVER\*\*: TIME UP !!! Terminating server... \n");

printf("\*\*SERVER\*\*: server ends\n");

exit(0);

}

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

{

signal(SIGALRM, handle\_sigalrm);

if (argc != 3) {

printf("Usage: %s <port-number> <time-duration>\n", argv[0]);

return -1;

} else {

printf("\n\*\*SERVER\*\*: Binding to Port: %s, Wait time: %s\n", argv[1], argv[2]);

}

int port = atoi(argv[1]);

int time\_to\_live = atoi(argv[2]);

int curr\_pid = getpid();

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

printf("\n\*\*SERVER\*\*: 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("Port %d still in use...\n", port);

sleep(5);

}

}

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

printf("\n\*\*SERVER\*\*: 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\*\*SERVER\*\*: 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)

{

alarm(time\_to\_live);

printf("\n\*\*SERVER\*\* Server will stay for %d seconds then terminate if no client connected\n", time\_to\_live);

printf("\n\*\*SERVER\*\* Step 1: Waiting for a client...\n");

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

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

if(connfd == -1){

perror("accept");

return -1;

}

printf("\n\*\*SERVER\*\* Step 2: A client connected.\n");

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

alarm(0);//reset alarm

printf("\n\*\*SERVER\*\*: Wait time resetted");

ticks = time(NULL);

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

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

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

perror("write\n");

return -1;

}

close(connfd);

printf("\n\*\*SERVER\*\* Step 3: Client disconnected \n\n");

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

sleep(1);

}

}

List Client2.c

{cslinux2:~/week12Lab2} cat Client2.c

// Client2.c

#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>

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

{

if (argc != 3) {

printf("Usage: %s <IPv4-address> <port-number>\n", argv[0]);

return -1;

} else {

printf("\n\*\*CLIENT\*\*: Connecting to IP: %s Port: %s\n", argv[1], argv[2]);

}

system("echo server; date; hostname; whoami; ps; ls -l");

int port = atoi(argv[2]);

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\*\*CLIENT\*\*: 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)

{

perror("\n\*\*CLIENT\*\*: Invalid address / Address not supported\n");

return 1;

}

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

printf("\n\*\*CLIENT\*\* Step 1: connecting to %s Port# = %d \n", argv[1], port);

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

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

{

printf("\n\*\*CLIENT\*\*: Error - Connect Failed \n");

return 1;

}

printf("\n\*\*CLIENT\*\* Step 2: connected to Server2. \n");

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

printf("\n\*\*CLIENT\*\*: reading from Server2. \n");

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

{

recvBuff[n] = 0;

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

{

printf("\n\*\*CLIENT\*\*: Error - Fputs error\n");

}

}

if(n < 0)

{

printf("\n\*\*CLIENT\*\*: Read error \n");

exit(0);

}

printf("\n\*\*CLIENT\*\* Step 3: Client disconnected\n");

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

printf("\n\*\*CLIENT\*\*: client ends! \n");

return 0;

}

5. Compile and check before running

{cslinux2:~/week12Lab2} gcc Server2.c -o Server2

{cslinux2:~/week12Lab2} gcc Client2.c -o Client2

{cslinux2:~/week12Lab2} ps

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8605 pts/0 00:00:00 ps

{cslinux2:~/week12Lab2} netstat -aont | grep 22719

No server process running and the port is free

6. Test Cases

#CASE 1

{cslinux2:~/week12Lab2} ./Server2 22719 10 &

[1] 8638

{cslinux2:~/week12Lab2}

\*\*SERVER\*\*: Binding to Port: 22719, Wait time: 10

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

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8638 pts/0 00:00:00 Server2

8645 pts/0 00:00:00 ps

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

\*\*SERVER\*\* Server will stay for 10 seconds then terminate if no client connected

\*\*SERVER\*\* Step 1: Waiting for a client...

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8638 pts/0 00:00:00 Server2

8649 pts/0 00:00:00 sh

8650 pts/0 00:00:00 ps

\*\*SERVER\*\*: TIME UP !!! Terminating server...

\*\*SERVER\*\*: server ends

Server terminated after 10 seconds of waiting

\*Checking if the process is terminated and port is freed

ps

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8679 pts/0 00:00:00 ps

[1]+ Done ./Server2 22719 10

{cslinux2:~/week12Lab2} netstat -aont | grep 22719

#CASE 2

{cslinux2:~/week12Lab2} ./Server2 22719 120 &

[1] 8741

{cslinux2:~/week12Lab2}

\*\*SERVER\*\*: Binding to Port: 22719, Wait time: 120

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

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8741 pts/0 00:00:00 Server2

8748 pts/0 00:00:00 ps

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

\*\*SERVER\*\* Server will stay for 120 seconds then terminate if no client connected

\*\*SERVER\*\* Step 1: Waiting for a client...

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8741 pts/0 00:00:00 Server2

8752 pts/0 00:00:00 sh

8753 pts/0 00:00:00 ps

hostname -i

10.176.92.16

\*check server address

{cslinux2:~/week12Lab2} ./Client2 10.176.92.16 22719

\*\*CLIENT\*\*: Connecting to IP: 10.176.92.16 Port: 22719

server

Wed Nov 6 17:04:59 CST 2024

cslinux2.utdallas.edu

txn200004

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8741 pts/0 00:00:00 Server2

8782 pts/0 00:00:00 Client2

8783 pts/0 00:00:00 sh

8787 pts/0 00:00:00 ps

total 96

-rwx--x--x 1 txn200004 se 13104 Nov 6 17:02 Client2

-rw------- 1 txn200004 se 2226 Nov 6 17:00 Client2.c

-rwx--x--x 1 txn200004 se 13928 Nov 6 17:02 Server2

-rw------- 1 txn200004 se 4938 Nov 6 17:00 Server2.c

\*\*CLIENT\*\* Step 1: connecting to 10.176.92.16 Port# = 22719

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8741 pts/0 00:00:00 Server2

8782 pts/0 00:00:00 Client2

8789 pts/0 00:00:00 sh

8790 pts/0 00:00:00 ps

\*\*CLIENT\*\* Step 2: connected to Server2.

\*\*SERVER\*\* Step 2: A client connected.

PID TTY TIME CMD

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8180 pts/0 00:00:00 bash

8741 pts/0 00:00:00 Server2

8741 pts/0 00:00:00 Server2

8782 pts/0 00:00:00 Client2

8782 pts/0 00:00:00 Client2

8795 pts/0 00:00:00 sh

8795 pts/0 00:00:00 sh

8796 pts/0 00:00:00 sh

8796 pts/0 00:00:00 sh

8797 pts/0 00:00:00 ps

8797 pts/0 00:00:00 ps

8798 pts/0 00:00:00 ps

8798 pts/0 00:00:00 ps

tcp 0 0 10.176.92.16:36428 10.176.92.16:22719 ESTABLISHED off (0.00/0/0)

tcp 0 0 10.176.92.16:22719 10.176.92.16:36428 ESTABLISHED off (0.00/0/0)

\*\*SERVER\*\*: Wait time resetted

\*\*SERVER\*\*: Writing to client...

\*\*SERVER\*\* Step 3: Client disconnected

tcp 0 0 10.176.92.16:36428 10.176.92.16:22719 ESTABLISHED off (0.00/0/0)

tcp 0 0 10.176.92.16:22719 10.176.92.16:36428 ESTABLISHED off (0.00/0/0)

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

\*\*CLIENT\*\*: reading from Server2.

Wed Nov 6 17:04:59 2024

\*\*CLIENT\*\* Step 3: Client disconnected

8741 pts/0 00:00:00 Server2

8782 pts/0 00:00:00 Client2

8807 pts/0 00:00:00 sh

8808 pts/0 00:00:00 ps

tcp 0 0 10.176.92.16:36428 10.176.92.16:22719 CLOSE\_WAIT off (0.00/0/0)

tcp 0 1 10.176.92.16:22719 10.176.92.16:36428 FIN\_WAIT1 on (0.16/0/0)

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8741 pts/0 00:00:00 Server2

8782 pts/0 00:00:00 Client2

8809 pts/0 00:00:00 sh

8810 pts/0 00:00:00 ps

tcp 0 0 10.176.92.16:36428 10.176.92.16:22719 CLOSE\_WAIT off (0.00/0/0)

tcp 0 0 10.176.92.16:22719 10.176.92.16:36428 FIN\_WAIT2 timewait (59.76/0/0)

\*\*CLIENT\*\*: client ends!

{cslinux2:~/week12Lab2}

\*\*SERVER\*\* Server will stay for 120 seconds then terminate if no client connected

\*\*SERVER\*\* Step 1: Waiting for a client...

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8741 pts/0 00:00:00 Server2

8821 pts/0 00:00:00 sh

8822 pts/0 00:00:00 ps

tcp 0 0 10.176.92.16:22719 10.176.92.16:36428 TIME\_WAIT timewait (59.19/0/0)

kill 8741 \*kill the server

\*Checking if the server is terminated and the port is freed

{cslinux2:~/week12Lab2} ps

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8853 pts/0 00:00:00 ps

[1]+ Terminated ./Server2 22719 120

{cslinux2:~/week12Lab2} netstat -aont | grep 22719

#CASE 3

{cslinux2:~/week12Lab2} ./Server2 22719 30 &

[1] 8968

{cslinux2:~/week12Lab2}

\*\*SERVER\*\*: Binding to Port: 22719, Wait time: 30

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

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8968 pts/0 00:00:00 Server2

8975 pts/0 00:00:00 ps

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

\*\*SERVER\*\* Server will stay for 30 seconds then terminate if no client connected

\*\*SERVER\*\* Step 1: Waiting for a client...

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8968 pts/0 00:00:00 Server2

8979 pts/0 00:00:00 sh

8980 pts/0 00:00:00 ps

./Client2 10.176.92.16 22719

\*\*CLIENT\*\*: Connecting to IP: 10.176.92.16 Port: 22719

server

Wed Nov 6 17:07:23 CST 2024

cslinux2.utdallas.edu

txn200004

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8968 pts/0 00:00:00 Server2

8995 pts/0 00:00:00 Client2

8996 pts/0 00:00:00 sh

9000 pts/0 00:00:00 ps

total 96

-rwx--x--x 1 txn200004 se 13104 Nov 6 17:02 Client2

-rw------- 1 txn200004 se 2226 Nov 6 17:00 Client2.c

-rwx--x--x 1 txn200004 se 13928 Nov 6 17:02 Server2

-rw------- 1 txn200004 se 4938 Nov 6 17:00 Server2.c

\*\*CLIENT\*\* Step 1: connecting to 10.176.92.16 Port# = 22719

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8968 pts/0 00:00:00 Server2

8995 pts/0 00:00:00 Client2

9003 pts/0 00:00:00 sh

9004 pts/0 00:00:00 ps

\*\*CLIENT\*\* Step 2: connected to Server2.

\*\*SERVER\*\* Step 2: A client connected.

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

PID TTY TIME CMD

8968 pts/0 00:00:00 Server2

8180 pts/0 00:00:00 bash

8995 pts/0 00:00:00 Client2

9011 pts/0 00:00:00 sh

9012 pts/0 00:00:00 sh

9013 pts/0 00:00:00 ps

9014 pts/0 00:00:00 ps

8968 pts/0 00:00:00 Server2

8995 pts/0 00:00:00 Client2

9011 pts/0 00:00:00 sh

9012 pts/0 00:00:00 sh

9013 pts/0 00:00:00 ps

9014 pts/0 00:00:00 ps

tcp 0 0 10.176.92.16:22719 10.176.92.16:36432 ESTABLISHED off (0.00/0/0)

tcp 0 0 10.176.92.16:36432 10.176.92.16:22719 ESTABLISHED off (0.00/0/0)

tcp 0 0 10.176.92.16:22719 10.176.92.16:36432 ESTABLISHED off (0.00/0/0)

tcp 0 0 10.176.92.16:36432 10.176.92.16:22719 ESTABLISHED off (0.00/0/0)

\*\*CLIENT\*\*: reading from Server2.

Wed Nov 6 17:07:25 2024

\*\*SERVER\*\*: Wait time resetted

\*\*SERVER\*\*: Writing to client...

\*\*SERVER\*\* Step 3: Client disconnected

\*\*CLIENT\*\* Step 3: Client disconnected

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8968 pts/0 00:00:00 Server2

8995 pts/0 00:00:00 Client2

PID TTY TIME CMD

9024 pts/0 00:00:00 sh

9025 pts/0 00:00:00 sh

8180 pts/0 00:00:00 bash

9026 pts/0 00:00:00 ps

9027 pts/0 00:00:00 ps

8968 pts/0 00:00:00 Server2

8995 pts/0 00:00:00 Client2

9024 pts/0 00:00:00 sh

9025 pts/0 00:00:00 sh

9026 pts/0 00:00:00 ps

9027 pts/0 00:00:00 ps

tcp 0 0 10.176.92.16:22719 10.176.92.16:36432 FIN\_WAIT2 timewait (59.61/0/0)

tcp 1 0 10.176.92.16:36432 10.176.92.16:22719 CLOSE\_WAIT off (0.00/0/0)

\*\*CLIENT\*\*: client ends!

{cslinux2:~/week12Lab2} tcp 0 0 10.176.92.16:22719 10.176.92.16:36432 FIN\_WAIT2 timewait (59.58/0/0)

tcp 1 0 10.176.92.16:36432 10.176.92.16:22719 CLOSE\_WAIT off (0.00/0/0)

\*\*SERVER\*\* Server will stay for 30 seconds then terminate if no client connected

\*\*SERVER\*\* Step 1: Waiting for a client...

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

8968 pts/0 00:00:00 Server2

9038 pts/0 00:00:00 sh

9039 pts/0 00:00:00 ps

tcp 0 0 10.176.92.16:22719 10.176.92.16:36432 TIME\_WAIT timewait (58.52/0/0)

\*\*SERVER\*\*: TIME UP !!! Terminating server...

\*\*SERVER\*\*: server ends

\*Checking if the server is terminated and the port is freed

ps

PID TTY TIME CMD

8180 pts/0 00:00:00 bash

9088 pts/0 00:00:00 ps

[1]+ Done ./Server2 22719 30

{cslinux2:~/week12Lab2} netstat -aont | grep 22719

End Lab and Exit

{cslinux2:~/week12Lab2} date

Wed Nov 6 17:09:22 CST 2024

{cslinux2:~/week12Lab2} ls -l

total 96

-rwx--x--x 1 txn200004 se 13104 Nov 6 17:02 Client2

-rw------- 1 txn200004 se 2226 Nov 6 17:00 Client2.c

-rwx--x--x 1 txn200004 se 13928 Nov 6 17:02 Server2

-rw------- 1 txn200004 se 4938 Nov 6 17:00 Server2.c

{cslinux2:~/week12Lab2} 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:~/week12Lab2} exit

logout

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

Session stopped

- Press <Return> to exit tab

- Press R to restart session

- Press S to save terminal output to file