**$ # cs360lab4**

**$ # Makiah Heinzmann & Taiya Williams**

**$ # 11442136 11644614**

**$ # server running:**

$ sudo ./server.bin 192.168.0.109

Initializing server

Server host info:

hostname=192.168.0.109

IP=192.168.0.109

Creating socket

Assigning name to socket

Getting port number from kernel

port=58210

Server Initialized

server: changed root to current directory

server: released root privileges

server: accepting new connections . . .

server: accepted a client:

IP=192.168.0.112 port=58922

in parent process

server: waiting for request from client . . .

server: accepting new connections . . .

server: read n=256 bytes:

ls

server: waiting for request from client . . .

server: read n=256 bytes:

pwd

sending: /

server: waiting for request from client . . .

server: read n=256 bytes:

cat clientfile1

server: waiting for request from client . . .

server: read n=256 bytes:

cat serverfile1

sending total file length: 20 bytes

wrote n=20 bytes to client, remaining length=0

server: waiting for request from client . . .

server: read n=256 bytes:

ls

server: waiting for request from client . . .

server: read n=256 bytes:

get serverfile1

sending total file length: 20 bytes

wrote n=20 bytes to client, remaining length=0

server: waiting for request from client . . .

server: read n=256 bytes:

rm serverfile1

server: waiting for request from client . . .

server: read n=256 bytes:

ls

server: waiting for request from client . . .

server: read n=256 bytes:

put serverfile1

client: ready to send file

Total File Length: 20 bytes:

write n=20 bytes to file=serverfile1, remaining length=0

server: waiting for request from client . . .

server: read n=256 bytes:

ls

server: waiting for request from client . . .

server: accepted a client:

IP=192.168.0.109 port=58221

in parent process

server: accepting new connections . . .

server: waiting for request from client . . .

server: read n=256 bytes:

put clientfile1

client: ready to send file

Total File Length: 14 bytes:

write n=14 bytes to file=clientfile1, remaining length=0

server: waiting for request from client . . .

server: read n=256 bytes:

rm clientfile1

server: waiting for request from client . . .

server: read n=256 bytes:

mkdir directory

directory 0755

server: waiting for request from client . . .

server: read n=256 bytes:

ls

server: waiting for request from client . . .

server: read n=256 bytes:

cd directory

server: waiting for request from client . . .

server: read n=256 bytes:

pwd

sending: /directory

server: waiting for request from client . . .

server: read n=256 bytes:

cd ../

server: waiting for request from client . . .

server: read n=256 bytes:

pwd

sending: /

server: waiting for request from client . . .

server: read n=256 bytes:

rmdir directory

server: waiting for request from client . . .

server: read n=256 bytes:

ls

server: waiting for request from client . . .

server: read n=256 bytes:

WOOOOOT

server: waiting for request from client . . .

server: client disconnected

**$ # client running**

$ ./client.bin 192.168.0.109 58210

Initializing client

Creating TCP socket

Connecting to server

connected to

hostname=192.168.0.109 IP=192.168.0.109 port=58210

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lls

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:50:26 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:48:09 2020 ..

-xwrxwrxw 1 1000 1000 22744 Thu Mar 5 15:45:31 2020 client.bin

-xwrxwrxw 1 1000 1000 14 Thu Mar 5 15:20:02 2020 clientfile1

-xwrxwrxw 1 1000 1000 0 Thu Mar 5 15:50:26 2020 output.txt

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : ls

client: wrote n=256 bytes:

ls

Server Response:

Permissions Links Group Owner Size Date Name

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:47:41 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:47:41 2020 ..

-xwrxwrxw 1 1000 1000 23120 Thu Mar 5 23:45:11 2020 server.bin

-xwrxwrxw 1 1000 1000 20 Thu Mar 5 23:44:48 2020 serverfile1

-xwrxwrxw 1 1000 1000 64 Thu Mar 5 23:47:44 2020 serverfile2

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : p lpwd

/mnt/c/Users/Tai/Documents/GitHub/CS360-Shared/Lab04/clienthome

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : pwd

client: wrote n=256 bytes:

pwd

Server Response:

/

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : cat clientfile1

client: wrote n=256 bytes:

cat clientfile1

Server Response:

Error: could not open file [ clientfile1 ] for reading.

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lcat clientfile1

thisisinafile

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : cat sere verfile1

client: wrote n=256 bytes:

cat serverfile1

Server Response:

file found

Total File Length: 20 bytes:

this is a file

yup

finished transmission

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : c m lmkdir directory

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : cd lc lls

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:52:53 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:48:09 2020 ..

-xwrxwrxw 1 1000 1000 22744 Thu Mar 5 15:45:31 2020 client.bin

-xwrxwrxw 1 1000 1000 14 Thu Mar 5 15:20:02 2020 clientfile1

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:52:53 2020 directory

-xwrxwrxw 1 1000 1000 0 Thu Mar 5 15:50:26 2020 output.txt

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lcd directr ory

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lpwd

/mnt/c/Users/Tai/Documents/GitHub/CS360-Shared/Lab04/clienthome/directory

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lcd ../

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lls lpwd

/mnt/c/Users/Tai/Documents/GitHub/CS360-Shared/Lab04/clienthome

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : r lrmdir directory

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lls

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:53:30 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:48:09 2020 ..

-xwrxwrxw 1 1000 1000 22744 Thu Mar 5 15:45:31 2020 client.bin

-xwrxwrxw 1 1000 1000 14 Thu Mar 5 15:20:02 2020 clientfile1

-xwrxwrxw 1 1000 1000 0 Thu Mar 5 15:50:26 2020 output.txt

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : ls

client: wrote n=256 bytes:

ls

Server Response:

Permissions Links Group Owner Size Date Name

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:47:41 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:47:41 2020 ..

-xwrxwrxw 1 1000 1000 23120 Thu Mar 5 23:45:11 2020 server.bin

-xwrxwrxw 1 1000 1000 20 Thu Mar 5 23:44:48 2020 serverfile1

-xwrxwrxw 1 1000 1000 64 Thu Mar 5 23:47:44 2020 serverfile2

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : get serverfile1

client: wrote n=256 bytes:

get serverfile1

Server Response:

file found

Total File Length: 20 bytes:

wrote n=20 bytes to file=serverfile1, remaining length=20

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lls

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:53:54 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:48:09 2020 ..

-xwrxwrxw 1 1000 1000 22744 Thu Mar 5 15:45:31 2020 client.bin

-xwrxwrxw 1 1000 1000 14 Thu Mar 5 15:20:02 2020 clientfile1

-xwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:53:33 2020 output.txt

-xwrxwrxw 1 1000 1000 20 Thu Mar 5 15:53:54 2020 serverfile1

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : rm serverfile1

client: wrote n=256 bytes:

rm serverfile1

Server Response:

Successfully removed filed [ serverfile1 ].

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : ls

client: wrote n=256 bytes:

ls

Server Response:

Permissions Links Group Owner Size Date Name

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:53:58 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:53:58 2020 ..

-xwrxwrxw 1 1000 1000 23120 Thu Mar 5 23:45:11 2020 server.bin

-xwrxwrxw 1 1000 1000 64 Thu Mar 5 23:47:44 2020 serverfile2

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : put serverfile1

client: wrote n=256 bytes:

put serverfile1

server: opened file for writing

sending total file length: 20 bytes

wrote n=20 bytes to client, remaining length=0

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lls

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:53:54 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:48:09 2020 ..

-xwrxwrxw 1 1000 1000 22744 Thu Mar 5 15:45:31 2020 client.bin

-xwrxwrxw 1 1000 1000 14 Thu Mar 5 15:20:02 2020 clientfile1

-xwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:53:33 2020 output.txt

-xwrxwrxw 1 1000 1000 20 Thu Mar 5 15:53:54 2020 serverfile1

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : ls

client: wrote n=256 bytes:

ls

Server Response:

Permissions Links Group Owner Size Date Name

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:54:15 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:54:15 2020 ..

-xwrxwrxw 1 1000 1000 23120 Thu Mar 5 23:45:11 2020 server.bin

-xwrxwrxw 1 1000 1000 20 Thu Mar 5 23:54:17 2020 serverfile1

-xwrxwrxw 1 1000 1000 64 Thu Mar 5 23:47:44 2020 serverfile2

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : rm serverfile1

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : put clientfile1

client: wrote n=256 bytes:

put clientfile1

server: opened file for writing

sending total file length: 14 bytes

wrote n=14 bytes to client, remaining length=0

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : rm clientfile1

client: wrote n=256 bytes:

rm clientfile1

Server Response:

Successfully removed filed [ clientfile1 ].

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : mkdir directory

client: wrote n=256 bytes:

mkdir directory

Server Response:

Created directory [ directory ].

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : ls

client: wrote n=256 bytes:

ls

Server Response:

Permissions Links Group Owner Size Date Name

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:55:12 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:55:12 2020 ..

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:55:12 2020 directory

-xwrxwrxw 1 1000 1000 23120 Thu Mar 5 23:45:11 2020 server.bin

-xwrxwrxw 1 1000 1000 20 Thu Mar 5 23:54:17 2020 serverfile1

-xwrxwrxw 1 1000 1000 64 Thu Mar 5 23:47:44 2020 serverfile2

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : cd directory

client: wrote n=256 bytes:

cd directory

Server Response:

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : pwd

client: wrote n=256 bytes:

pwd

Server Response:

/directory

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : cd ../

client: wrote n=256 bytes:

cd ../

Server Response:

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : pwd

client: wrote n=256 bytes:

pwd

Server Response:

/

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : rmdir directory

client: wrote n=256 bytes:

rmdir directory

Server Response:

Successfully removed directory [ directory ].

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : lls

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:54:33 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 15:48:09 2020 ..

-xwrxwrxw 1 1000 1000 22744 Thu Mar 5 15:45:31 2020 client.bin

-xwrxwrxw 1 1000 1000 14 Thu Mar 5 15:20:02 2020 clientfile1

-xwrxwrxw 1 1000 1000 8192 Thu Mar 5 15:54:51 2020 output.txt

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : ls

client: wrote n=256 bytes:

ls

Server Response:

Permissions Links Group Owner Size Date Name

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:55:48 2020 .

dxwrxwrxw 1 1000 1000 4096 Thu Mar 5 23:55:48 2020 ..

-xwrxwrxw 1 1000 1000 23120 Thu Mar 5 23:45:11 2020 server.bin

-xwrxwrxw 1 1000 1000 20 Thu Mar 5 23:54:17 2020 serverfile1

-xwrxwrxw 1 1000 1000 64 Thu Mar 5 23:47:44 2020 serverfile2

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : WOOOOOT

client: wrote n=256 bytes:

WOOOOOT

server:

server: command not found

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : ^C

**# $ server running – multiple clients**

$ sudo ./server.bin

[sudo] password for kiah:

Initializing server

Server host info:

hostname=localhost

IP=127.0.0.1

Creating socket

Assigning name to socket

Getting port number from kernel

port=58425

Server Initialized

server: changed root to current directory

server: released root privileges

server: accepting new connections . . .

server: accepted a client:

IP=127.0.0.1 port=58431

in parent process

server: waiting for request from client . . .

server: accepting new connections . . .

server: accepted a client:

IP=127.0.0.1 port=58432

in parent process

server: waiting for request from client . . .

server: accepting new connections . . .

server: read n=256 bytes:

quit

server: client quit program

server: read n=256 bytes:

quit

server: client quit program

**$ # multiple clients running on one server**

**$ # client 1**

$ ./client.bin localhost 58425

Initializing client

Creating TCP socket

Connecting to server

connected to

hostname=localhost IP=127.0.0.1 port=58425

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : quit

**$ # client 2**

$ ./client.bin localhost 58425

Initializing client

Creating TCP socket

Connecting to server

connected to

hostname=localhost IP=127.0.0.1 port=58425

| get put cat ls cd pwd mkdir rmdir rm |

| lcat lls lcd lpwd lmkdir lrmdir lrm |

input a line : quit

**$ cat build**

#! /bin/bash

touch serverhome/server.bin

rm serverhome/server.bin

gcc -o serverhome/server.bin server.c

sudo chown root:root serverhome/server.bin

sudo chmod u+s serverhome/server.bin

touch clienthome/client.bin

rm clienthome/client.bin

gcc -o clienthome/client.bin client.c

**$ cat client.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <netdb.h>

#include <sys/socket.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <dirent.h>

#include <sys/stat.h>

#include <stdbool.h>

#include <time.h>

#include <arpa/inet.h>

#define LINEMAX 256

struct hostent \* host\_entry;

struct sockaddr\_in server\_addr;

char \* permAvailable = "xwrxwrxwr-------";

char \* permRestricted = "----------------";

struct in\_addr server\_ip;

int server\_socket, server\_port;

void lsFile(char \* fileStr);

void lsDir(char \* dirStr);

void client\_init(char \* argv[]) {

printf("Initializing client\n");

host\_entry = gethostbyname(argv[1]);

if (host\_entry == NULL) {

printf("unknown host %s\n", argv[1]);

exit(2);

}

server\_ip = \*((struct in\_addr \*) host\_entry->h\_addr\_list[0]);

server\_port = atoi(argv[2]);

printf("Creating TCP socket\n");

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

if (server\_socket < 0) {

printf("failed to create socket\n");

exit(3);

}

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

server\_addr.sin\_addr.s\_addr = server\_ip.s\_addr;

server\_addr.sin\_port = htons(server\_port);

printf("Connecting to server\n");

if (connect(server\_socket, (struct sockaddr \*) & server\_addr,

sizeof(server\_addr)) < 0) {

printf("connection failed\n");

exit(4);

}

printf("connected to \n");

char ip[24];

inet\_ntop(AF\_INET, (struct in\_addr \*) host\_entry->h\_addr\_list[0], ip, sizeof(ip));

printf(" hostname=%s IP=%s port=%d\n", host\_entry->h\_name,

ip, server\_port);

}

void cat(char \* line) {

strtok(line, " ");

char \* file = strtok(NULL, " ");

if (access(file, F\_OK) >= 0) {

char buffer[1024];

int fdesc = open(file, O\_RDONLY);

if (fdesc != -1) {

while (read(fdesc, &buffer, 1024) > 0) {

printf("%s", buffer);

bzero(buffer, 1024);

}

printf("\n");

close(fdesc);

} else {

printf("Error: could not open file [ %s ] for reading.\n", file);

}

} else {

printf("Error: no such file [ %s ] was found.\n", file);

}

}

void ls(char \* line) {

strtok(line, " ");

char \* paramStr = strtok(NULL, " ");

if (paramStr != NULL) {

struct stat \* stats = (struct stat \*) malloc(sizeof(struct stat));

if (lstat(paramStr, stats) == 0) {

printf("Permissions Links Group Owner Size Date Name\n");

if (S\_ISDIR(stats->st\_mode)) {

lsDir(paramStr);

} else {

lsFile(paramStr);

}

} else {

printf("Error: no such file or directory [ %s ] found.\n", paramStr);

}

free(stats);

} else {

lsDir("./");

}

}

void lsDir(char \* dirStr) {

DIR \* dir = opendir(dirStr);

if (dir != NULL) {

struct dirent \* treebeard = readdir(dir);

// "The world is changing:

// I feel it in the water,

// I feel it in the earth,

// and I smell it in the air."

// Treebeard, The Two Towers, J. R. R. Tolkien.

char path[4356]; //max path length + entry name size = 4096 + 260 = 4356 characters

while(treebeard != NULL) {

bzero(path, 4356);

strcat(path, dirStr);

strcat(path, treebeard->d\_name);

lsFile(path);

treebeard = readdir(dir);

}

closedir(dir);

} else {

printf("Error: no such directory [ %s ] found.\n", dirStr);

}

}

void lsFile(char \* fileStr) {

if (access(fileStr, F\_OK) == 0) {

struct stat \* stats = (struct stat \*) malloc(sizeof(struct stat));

lstat(fileStr, stats);

char type = '0'; //Other/unknown type

if (S\_ISDIR(stats->st\_mode)) {

type = 'd';

} else if (S\_ISREG(stats->st\_mode)) {

type = '-';

} /\*else if (S\_ISLINK(stats->st\_mode)) {

type = 'l';

}\*/

printf("%c", type);

for (int i = 0; i < 8; i++) {

if (stats->st\_mode & (1 << i)) { // print r | w | x

printf("%c", permAvailable[i]);

} else {

printf("%c", permRestricted[i]);

}

}

char \* fileTime = ctime(&(stats->st\_ctime));

fileTime[strlen(fileTime) - 1] = '\0';

// Permissions Links Group Owner Size Date Name

printf(" %ld", stats->st\_nlink);

printf(" %d", stats->st\_gid);

printf(" %d", stats->st\_uid);

printf(" %ld", stats->st\_size);

printf(" %s", fileTime);

printf(" %s\n", fileStr + 2);

free(stats);

} else {

printf("Error: no such file [ %s ] found.\n", fileStr);

}

}

void put(char \* line) {

char linecpy[LINEMAX + 1];

strcpy(linecpy, line);

strtok(linecpy, " ");

char \* file = strtok(NULL, " ");

if (access(file, F\_OK) >= 0) {

int fdesc = open(file, O\_RDONLY);

if (fdesc != -1) {

struct stat fstat;

stat(file, &fstat);

long length = fstat.st\_size;

read(server\_socket, line, LINEMAX);

printf("server: %s", line);

if (strncmp(line, "error", 5) == 0) {

close(fdesc);

return;

}

sprintf(line, "ready to send file\n");

write(server\_socket, line, LINEMAX);

write(server\_socket, &(fstat.st\_size), sizeof(long));

printf("sending total file length: %ld bytes\n", length);

int n;

while (length > LINEMAX) {

n = read(fdesc, line, LINEMAX);

length -= n;

write(server\_socket, line, LINEMAX);

printf("wrote n=%d bytes to client, remaining length=%ld\n",

n, length);

}

n = read(fdesc, line, LINEMAX);

length -= n;

write(server\_socket, line, n);

printf("wrote n=%d bytes to client, remaining length=%ld\n",

n, length);

close(fdesc);

} else {

printf("Error: could not open file [ %s ] for reading.\n", file);

}

} else {

printf("Error: could not open file [ %s ] for reading.\n", file);

}

}

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

int n;

char line[LINEMAX + 1];

if (argc < 3) {

printf("Required:\n cient.bin <<ServerName>> <<ServerPort>>\n");

exit(1);

}

client\_init(argv);

while (true) {

printf("| get put cat ls cd pwd mkdir rmdir rm |\n");

printf("| lcat lls lcd lpwd lmkdir lrmdir lrm |\n");

printf("input a line : ");

bzero(line, LINEMAX); // zero out line[ ]

fgets(line, LINEMAX, stdin); // get a line (end with \n) from stdin

line[strlen(line) - 1] = '\0';

if (strncmp(line, "quit", 4) == 0){

write(server\_socket, line, LINEMAX);

exit(0);

} else if (strncmp(line, "lcat", 3) == 0) {

cat(line);

} else if (strncmp(line, "lpwd", 4) == 0) {

char buffer[512];

getcwd(buffer, 512);

printf("%s\n", buffer);

} else if (strncmp(line, "lls", 3) == 0) {

ls(line);

} else if (strncmp(line, "lcd", 3) == 0) {

strtok(line, " ");

char \* dirpath = strtok(NULL, " ");

if (chdir(dirpath) != 0)

printf("error: could not find directory\n");

} else if (strncmp(line, "lmkdir", 6) == 0) {

strtok(line, " ");

char \* name = strtok(NULL, " ");

if (

name != NULL &&

opendir(name) == NULL

) {

mkdir(name, 0755);

} else {

printf("Error: could not create directory [ %s ].\n", name);

}

} else if (strncmp(line, "lrmdir", 6) == 0) {

strtok(line, " ");

char \* name = strtok(NULL, " ");

if (

name != NULL &&

opendir(name) != NULL

) {

rmdir(name);

} else {

printf("Error: could not remove directory [ %s ].\n", name);

}

} else if (strncmp(line, "lrm", 3) == 0) {

strtok(line, " ");

char \* name = strtok(NULL, " ");

if (

name != NULL &&

access(name, F\_OK) == 0

) {

remove(name);

} else {

printf("Error: could not remove file [ %s ].\n", name);

}

} else {

// Send ENTIRE line to server

n = write(server\_socket, line, LINEMAX);

printf("client: wrote n=%d bytes:\n %s\n", n, line);

if (

strncmp(line, "pwd", 3) == 0 ||

strncmp(line, "ls", 2) == 0 ||

strncmp(line, "mkdir", 5) == 0 ||

strncmp(line, "rmdir", 5) == 0 ||

strncmp(line, "rm", 2) == 0 ||

strncmp(line, "cd", 2) == 0) {

printf("\tServer Response:\n\n");

bzero(line, LINEMAX);

read(server\_socket, line, LINEMAX);

while (strcmp(line, "") != 0) {

printf("%s", line);

bzero(line, LINEMAX);

read(server\_socket, line, LINEMAX);

}

printf("\n");

} else if (strncmp(line, "cat", 3) == 0) {

read(server\_socket, line, LINEMAX);

printf("\tServer Response:\n %s", line);

if (strncmp(line, "Error", 5) == 0) {

read(server\_socket, line, LINEMAX);

printf("%s", line);

read(server\_socket, line, LINEMAX);

printf("%s\n", line);

} else {

printf("\n");

bzero(line, LINEMAX + 1);

long length = 0;

read(server\_socket, &length, sizeof(long));

printf("Total File Length: %ld bytes:\n\n", length);

int n;

while (length > LINEMAX) {

n = read(server\_socket, line, LINEMAX);

length -= n;

printf("%s", line);

}

n = read(server\_socket, line, length);

line[n] = '\0';

printf("%s\n\nfinished transmission\n", line);

}

read(server\_socket, line, LINEMAX);

} else if (strncmp(line, "get", 3) == 0) {

char linecpy[LINEMAX + 1];

strcpy(linecpy, line);

read(server\_socket, line, LINEMAX);

printf("\tServer Response:\n %s", line);

if (strncmp(line, "Error", 5) == 0) {

read(server\_socket, line, LINEMAX);

printf("%s", line);

read(server\_socket, line, LINEMAX);

printf("%s\n", line);

} else {

printf("\n");

bzero(line, LINEMAX + 1);

long length = 0;

read(server\_socket, &length, sizeof(long));

printf("Total File Length: %ld bytes:\n\n", length);

int n;

strtok(linecpy, " ");

char \* filename = strtok(NULL, " ");

int fd = open(filename, O\_WRONLY|O\_CREAT, 0644);

while (length > LINEMAX) {

n = read(server\_socket, line, LINEMAX);

length -= n;

write(fd, line, LINEMAX);

printf("wrote n=%d bytes to file=%s, remaining length=%ld\n",

n, filename, length);

}

n = read(server\_socket, line, length);

length -+ n;

write(fd, line, n);

printf("wrote n=%d bytes to file=%s, remaining length=%ld\n",

n, filename, length);

close(fd);

}

read(server\_socket, line, LINEMAX);

} else if (strncmp(line, "put", 3) == 0) {

put(line);

read(server\_socket, line, LINEMAX);

} else {

read(server\_socket, line, LINEMAX);

printf("server:\n %s\n", line);

read(server\_socket, line, LINEMAX);

}

}

}

}

**$ cat server.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <netdb.h>

#include <sys/socket.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <dirent.h>

#include <sys/stat.h>

#include <stdbool.h>

#include <time.h>

#include <arpa/inet.h>

#define LINEMAX 256

char \* permAvailable = "xwrxwrxwr-------";

char \* permRestricted = "----------------";

struct hostent \* host\_entry;

struct sockaddr\_in server\_addr, client\_addr, name\_addr;

int server\_socket, client\_socket, server\_port;

char cwd[4096];

char line[LINEMAX + 1];

void lsFile(char \* fileStr);

void lsDir(char \* dirStr);

void server\_init(char \* name) {

printf("Initializing server\n");

host\_entry = gethostbyname(name);

if (host\_entry == NULL) {

printf("unknown host\n");

exit(1);

}

printf("Server host info:\n");

printf(" hostname=%s\n", name);

char ip[16];

inet\_ntop(AF\_INET, (struct in\_addr \*) host\_entry->h\_addr\_list[0], ip, sizeof(ip));

printf(" IP=%s\n", ip);

// printf("IP=%s\n", inet\_ntoa(\*((struct in\_addr \*) host\_entry->h\_addr\_list[0])));

printf("Creating socket\n");

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

if (server\_socket < 0) {

printf("failed to create socket\n");

exit(2);

}

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

server\_addr.sin\_addr = \*((struct in\_addr \*) host\_entry->h\_addr\_list[0]);

// server\_addr.sin\_addr.s\_addr = \*(long \*) host\_entry->h\_addr\_list[0];

server\_addr.sin\_port = 0; // kernal will assign port number

printf("Assigning name to socket\n");

if (bind(server\_socket, (struct sockaddr \*) & server\_addr,

sizeof(server\_addr)) != 0) {

printf("failed to bind socket to address\n");

exit(3);

}

printf("Getting port number from kernel\n");

int len\_name\_addr = sizeof(name\_addr);

if (getsockname(server\_socket, (struct sockaddr \*) & name\_addr,

& len\_name\_addr) != 0) {

printf("failed getting socket name\n");

exit(4);

}

server\_port = ntohs(name\_addr.sin\_port);

printf(" port=%d\n", server\_port);

listen(server\_socket, 5);

printf("Server Initialized\n");

}

void get(char \* line) {

strtok(line, " ");

char \* file = strtok(NULL, " ");

if (access(file, F\_OK) >= 0) {

int fdesc = open(file, O\_RDONLY);

if (fdesc != -1) {

struct stat fstat;

stat(file, &fstat);

long length = fstat.st\_size;

sprintf(line, "file found\n");

write(client\_socket, line, LINEMAX);

write(client\_socket, &length, sizeof(long));

printf("sending total file length: %ld bytes\n", length);

int n;

while (length > LINEMAX) {

n = read(fdesc, line, LINEMAX);

length -= n;

write(client\_socket, line, LINEMAX);

printf("wrote n=%d bytes to client, remaining length=%ld\n",

n, length);

}

n = read(fdesc, line, LINEMAX);

length -= n;

write(client\_socket, line, n);

printf("wrote n=%d bytes to client, remaining length=%ld\n",

n, length);

close(fdesc);

} else {

write(client\_socket, "Error: could not open file [ ", LINEMAX);

write(client\_socket, file, LINEMAX);

write(client\_socket, " ] for reading.\n", LINEMAX);

}

} else {

write(client\_socket, "Error: could not open file [ ", LINEMAX);

write(client\_socket, file, LINEMAX);

write(client\_socket, " ] for reading.\n", LINEMAX);

}

}

void cat(char \* line) {

strtok(line, " ");

char \* file = strtok(NULL, " ");

if (access(file, F\_OK) >= 0) {

int fdesc = open(file, O\_RDONLY);

if (fdesc != -1) {

struct stat fstat;

stat(file, &fstat);

long length = fstat.st\_size;

sprintf(line, "file found\n");

write(client\_socket, line, LINEMAX);

write(client\_socket, &length, sizeof(long));

printf("sending total file length: %ld bytes\n", length);

int n;

while (length > LINEMAX) {

n = read(fdesc, line, LINEMAX);

length -= n;

write(client\_socket, line, LINEMAX);

printf("wrote n=%d bytes to client, remaining length=%ld\n",

n, length);

}

n = read(fdesc, line, LINEMAX);

length -= n;

write(client\_socket, line, n);

printf("wrote n=%d bytes to client, remaining length=%ld\n",

n, length);

close(fdesc);

} else {

write(client\_socket, "Error: could not open file [ ", LINEMAX);

write(client\_socket, file, LINEMAX);

write(client\_socket, " ] for reading.\n", LINEMAX);

}

} else {

write(client\_socket, "Error: could not open file [ ", LINEMAX);

write(client\_socket, file, LINEMAX);

write(client\_socket, " ] for reading.\n", LINEMAX);

}

}

void ls(char \* line) {

char lsarg[256];

strcpy(lsarg, line);

strtok(lsarg, " ");

char \* paramStr = strtok(NULL, " ");

if (paramStr != NULL) {

strncpy(lsarg, paramStr, LINEMAX);

struct stat \* stats = (struct stat \*) malloc(sizeof(struct stat));

write(client\_socket, "Permissions Links Group Owner Size Date Name\n", 46);

if (lstat(lsarg, stats) == 0) {

if (S\_ISDIR(stats->st\_mode)) {

lsDir(lsarg);

} else {

lsFile(lsarg);

}

} else {

write(client\_socket, "Error: no such file or directory [ ", LINEMAX);

write(client\_socket, lsarg, LINEMAX);

write(client\_socket, " ] found.\n", LINEMAX);

}

free(stats);

} else {

write(client\_socket, "Permissions Links Group Owner Size Date Name\n", LINEMAX);

lsDir("./");

}

}

void lsDir(char \* dirStr) {

DIR \* dir = opendir(dirStr);

if (dir != NULL) {

struct dirent \* treebeard = readdir(dir);

// "The world is changing:

// I feel it in the water,

// I feel it in the earth,

// and I smell it in the air."

// Treebeard, The Two Towers, J. R. R. Tolkien.

char path[4356]; //max path length + entry name size = 4096 + 260 = 4356 characters

while(treebeard != NULL) {

bzero(path, 4356);

strcat(path, dirStr);

strcat(path, treebeard->d\_name);

lsFile(path);

treebeard = readdir(dir);

}

} else {

write(client\_socket, "Error: no such directory [ ", LINEMAX);

write(client\_socket, dirStr, LINEMAX);

write(client\_socket, " ] found.\n", LINEMAX);

}

}

void lsFile(char \* fileStr) {

if (access(fileStr, F\_OK) == 0) {

struct stat \* stats = (struct stat \*) malloc(sizeof(struct stat));

lstat(fileStr, stats);

char permissions[10];

permissions[0] = 'd'; //Other/unknown type

if (S\_ISDIR(stats->st\_mode)) {

permissions[0] = 'd';

} else if (S\_ISREG(stats->st\_mode)) {

permissions[0] = '-';

} /\*else if (S\_ISLINK(stats->st\_mode)) {

permissions[0] = 'l';

}\*/

for (int i = 0; i < 8; i++) {

if (stats->st\_mode & (1 << i)) { // print r | w | x

permissions[i+1] = permAvailable[i];

} else {

permissions[i+1] = permRestricted[i];

}

}

char \* fileTime = ctime(&(stats->st\_ctime));

fileTime[strlen(fileTime) - 1] = '\0';

bzero(line, LINEMAX);

sprintf(

line,

"%s %ld %d %d %ld %s %s\n",

permissions,

stats->st\_nlink,

stats->st\_gid,

stats->st\_uid,

stats->st\_size,

fileTime,

fileStr + 2

);

write(client\_socket, line, LINEMAX);

free(stats);

} else {

write(client\_socket, "Error: no such file [ ", LINEMAX);

write(client\_socket, fileStr, LINEMAX);

write(client\_socket, " ] found.\n", LINEMAX);

}

}

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

char hostname[256];

int n;

if (argc < 2) {

strcpy(hostname, "localhost");

// gethostname(hostname, 256);

}

else

strncpy(hostname, argv[1], 255);

server\_init(hostname);

getcwd(cwd, 4096);

int changed = chroot(cwd);

if (changed != 0) {

printf("error: chroot failed\n");

exit(8);

}

chdir("/");

getcwd(cwd, 4096);

printf("server: changed root to current directory\n");

if (setgid(getgid()) == -1) {

printf("error: failed to release permissions\n");

exit(9);

}

if (setuid(getuid()) == -1) {

printf("error: failed to release permissions\n");

exit(10);

}

printf("server: released root privileges\n");

while (true) {

printf("server: accepting new connections . . .\n");

int len\_client\_addr = sizeof(client\_addr);

client\_socket = accept(server\_socket, (struct sockaddr \*) & client\_addr,

& len\_client\_addr);

if (client\_socket < 0) {

printf("server: error accepting new client\n");

exit(5);

}

printf("server: accepted a client:\n");

char ip[24];

inet\_ntop(AF\_INET, &client\_addr.sin\_addr, ip, sizeof(ip));

printf(" IP=%s port=%d\n", ip, ntohs(client\_addr.sin\_port));

if (fork()) { // parent

close(client\_socket);

printf("in parent process\n");

}

else {

while (true) { // processing loop

printf("server: waiting for request from client . . .\n");

n = read(client\_socket, line, LINEMAX);

if (n == 0) {

printf("server: client disconnected\n");

close(client\_socket);

exit(0);

}

printf("server: read n=%d bytes:\n %s\n", n, line);

if (strncmp(line, "pwd", 3) == 0) {

strncpy(line, cwd, LINEMAX);

printf("sending: %s\n", line);

write(client\_socket, line, LINEMAX);

} else if (strncmp(line, "ls", 2) == 0) {

ls(line);

} else if (strncmp(line, "cat", 3) == 0) {

cat(line);

} else if (!strncmp(line, "cd", 2)) {

strtok(line, " ");

char \* dirpath = strtok(NULL, " ");

if (chdir(dirpath) != 0)

write(client\_socket, "error: could not find directory\n", LINEMAX);

getcwd(cwd, 4096);

} else if (strncmp(line, "mkdir", 5) == 0) {

strtok(line, " ");

char \* name = strtok(NULL, " ");

printf("\t%s %s\n", name, "0755");

if (

name != NULL &&

opendir(name) == NULL

) {

mkdir(name, 0755);

write(client\_socket, "Created directory [ ", LINEMAX);

write(client\_socket, name, LINEMAX);

write(client\_socket, " ].\n", LINEMAX);

} else {

write(client\_socket, "Error: could not create directory [ ", LINEMAX);

write(client\_socket, name, LINEMAX);

write(client\_socket, " ].\n", LINEMAX);

}

} else if (!strncmp(line, "rmdir", 5)) {

strtok(line, " ");

char \* name = strtok(NULL, " ");

if (

name != NULL &&

opendir(name) != NULL

) {

rmdir(name);

write(client\_socket, "Successfully removed directory [ ", LINEMAX);

write(client\_socket, name, LINEMAX);

write(client\_socket, " ].\n", LINEMAX);

} else {

write(client\_socket, "Error: could not remove directory [ ", LINEMAX);

write(client\_socket, name, LINEMAX);

write(client\_socket, " ].\n", LINEMAX);

}

} else if (!strncmp(line, "rm", 2)) {

strtok(line, " ");

char \* name = strtok(NULL, " ");

if (

name != NULL &&

access(name, F\_OK) == 0

) {

remove(name);

write(client\_socket, "Successfully removed filed [ ", LINEMAX);

write(client\_socket, name, LINEMAX);

write(client\_socket, " ].\n", LINEMAX);

} else {

write(client\_socket, "Error: could not remove file [ ", LINEMAX);

write(client\_socket, name, LINEMAX);

write(client\_socket, " ].\n", LINEMAX);

}

} else if (!strncmp(line, "get", 3)) {

get(line);

} else if (!strncmp(line, "put", 3)) {

char linecpy[LINEMAX + 1];

strcpy(linecpy, line);

strtok(linecpy, " ");

char \* filename = strtok(NULL, " ");

if (access(filename, F\_OK) != 0) {

int fdesc = open(filename, O\_WRONLY|O\_CREAT, 0644);

if (fdesc != -1) {

sprintf(line, "opened file for writing\n");

write(client\_socket, line, LINEMAX);

read(client\_socket, line, LINEMAX);

printf("client: %s\n", line);

long length = 0;

read(client\_socket, &length, sizeof(long));

printf("Total File Length: %ld bytes:\n\n", length);

int n;

while (length > LINEMAX) {

n = read(client\_socket, line, LINEMAX);

length -= n;

write(fdesc, line, LINEMAX);

printf("wrote n=%d bytes to file=%s, reamaining length=%ld\n",

n, filename, length);

}

n = read(client\_socket, line, length);

write(fdesc, line, n);

length -= n;

printf("write n=%d bytes to file=%s, remaining length=%ld\n",

n, filename, length);

close(fdesc);

} else {

sprintf(line, "error: could not open file for writing\n");

write(client\_socket, line, LINEMAX);

}

} else {

sprintf(line, "error: file already exists\n");

write(client\_socket, line, LINEMAX);­­­­­­­

}

} else if (!strncmp(line, "quit", 4)) {

printf("server: client quit program\n");

close(client\_socket);

exit(0);

} else {

strcpy(line, "server: command not found\n");

write(client\_socket, line, LINEMAX);

}

write(client\_socket, "", LINEMAX);

}

}

}

}­