**Enoch Ikunda**

**COS350**

**PROG3: mysubmit.c**

**Prof: Bob Boothe**

**Note: You allowed me a little extension due to my poor internet**

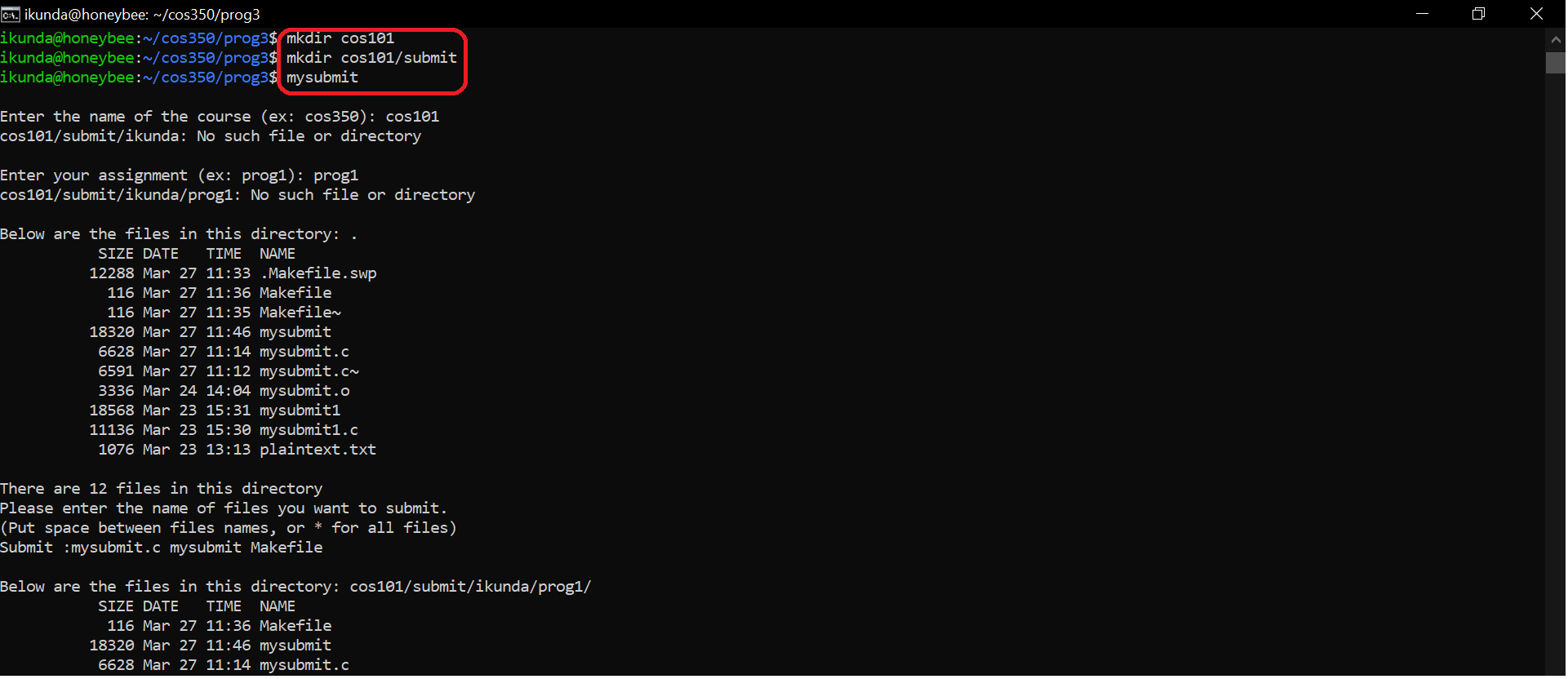
[could not print using a2ps]

**mkdir cos101** make a new empty directory for testing

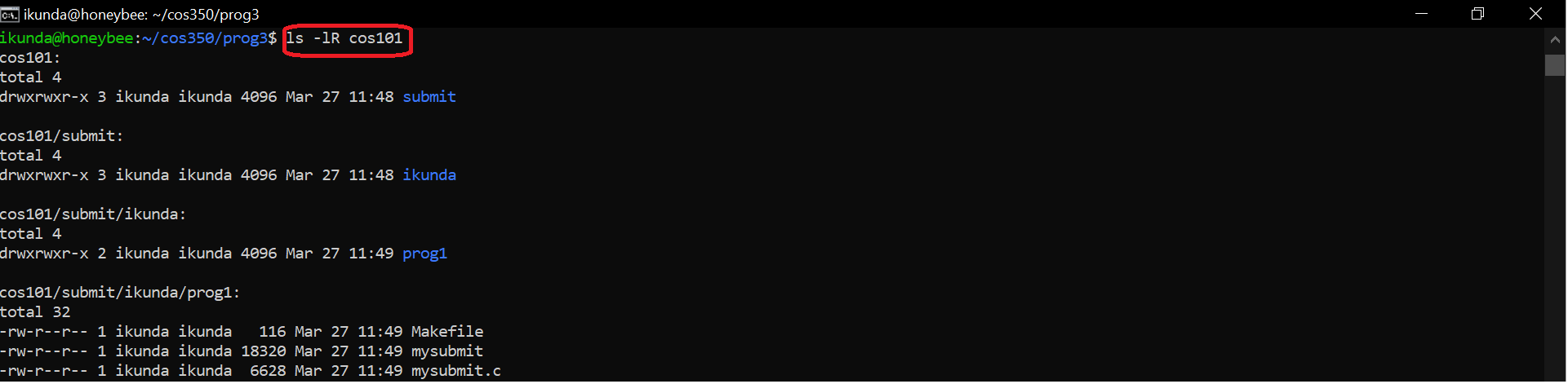
**mkdir cos101/submit** make the submit subdirectory

**mysubmit** run your program and submit to cos101 the three

files: mysubmit.c mysubmit Makefile

****

**ls –lR cos101** recursive listing to verify everything worked

****

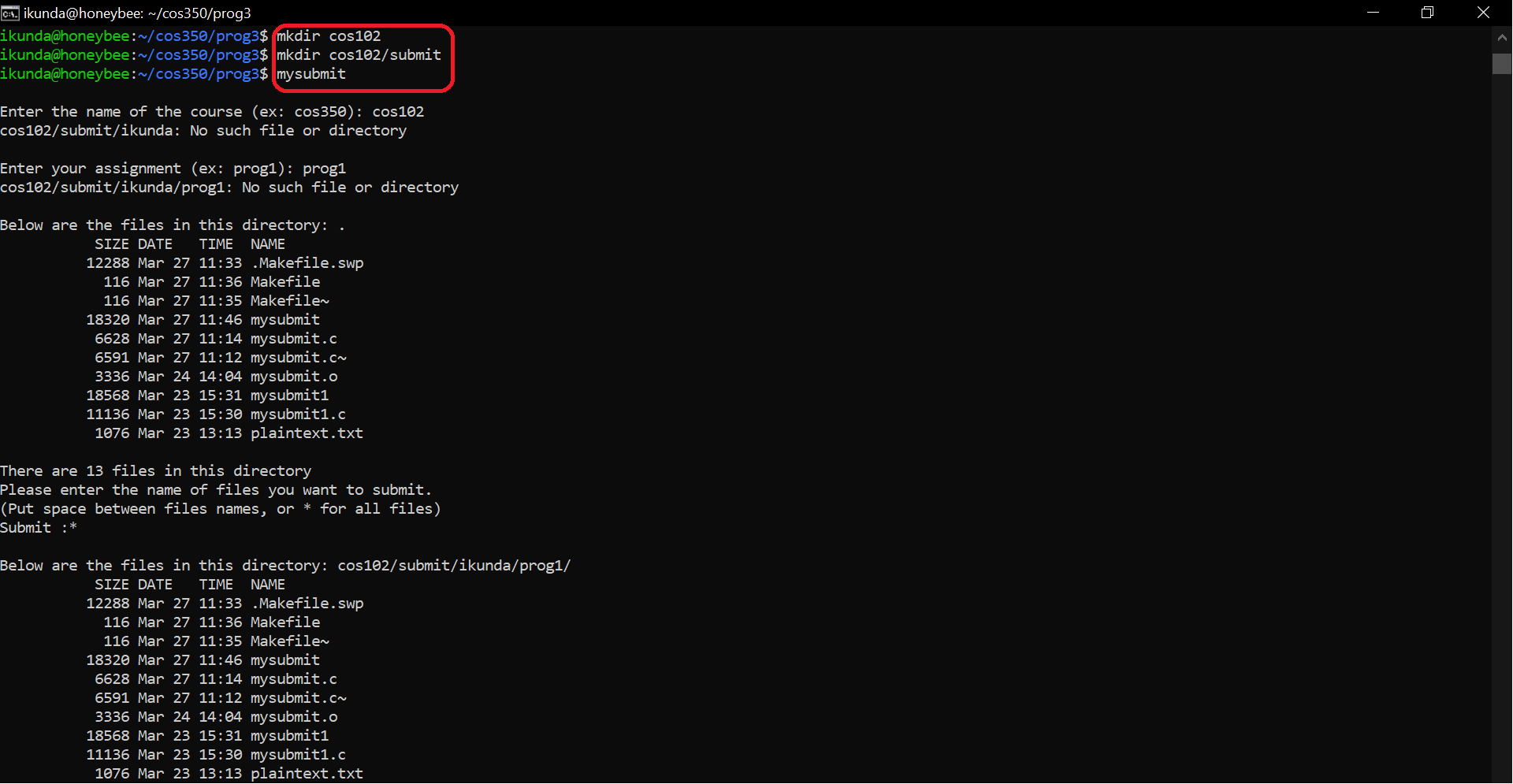
**mkdir cos102** make a new empty directory for testing

**mkdir cos102/submit** make the submit subdirectory

**mysubmit** run your program and submit to cos102,

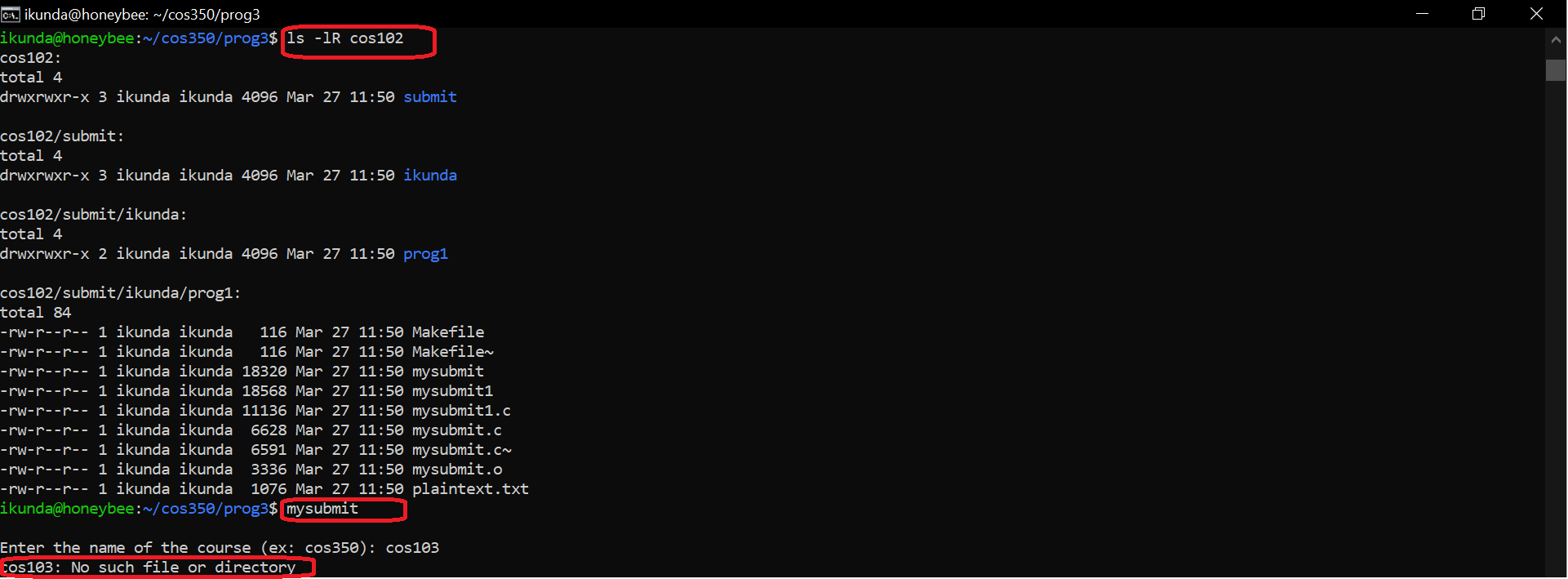
but this time just use \* to

submit everything

****

**ls –lR cos102** recursive listing to verify everything worked

**mysubmit** run it again but this time use a non-existent course directory cos103 to test error handling

****

/\*Prog3 by Enoch Ikunda

\* Prof: Bob Boothe

\* Purpose : list & transfer filest to a desired (but specific) folder

\* Action: no argument needed just fill in the prompts

\* note: The progrma can either take selected files or all the files from desired directory

\* Bug: only works in current directory

\*/

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <time.h>

#include <dirent.h>

#include <pwd.h>

#include <fcntl.h>

#define INPUTBUF 512

#define BUFFERSIZE 4096

#define COPYMODE 0644

char coursepointer [INPUTBUF];

char course [INPUTBUF];

char projectpath [INPUTBUF];

uid\_t id;

char \*user;

int verfydirexist (char \*dirname);

char \*get\_user(uid\_t uid);

int verifyfiles (char \*currentdir);

int displayfiles ();

void dostat(char \*);

void show\_file\_info(char \*, struct stat \*);

void transferfiles( char \*files, char \*path);

void pathupdated (char \*updatepath, char \*path, char \*submited);

void dotransfer (char \*newpath, char \*filesub);

void oops (char \*s1, char \*s2);

int main()

{

printf("\nEnter the name of the course (ex: cos350): ");

if(fgets(coursepointer,INPUTBUF,stdin))

{

coursepointer[strcspn(coursepointer,"\n")] = 0;

}

strcpy(course, coursepointer);

//1)\*\*\*\*the course and verify that directory exists\*\*\*\*

if(verfydirexist(course) == -1)

{

perror (course);

exit(0);

}

//2)\*\*\*\*Verify that it contains a submit directory\*\*\*\*

strcat(course,"/submit");

if(verfydirexist(course) == -1)

{

perror (course);

exit(0);

}

//3 & 4)\*\*\*\*\*\*\*Get the user's name from the system &

//If this user does not yet have a subdirectory within

//the submit directory, create one\*\*\*\*\*

strcat(course,"/");

id = getuid();

user = get\_user(id);

strcat(course,user);

if(verfydirexist(course) == -1)

{

perror (course);

mkdir(course, 0777);

}

//5)\*\*\*\*\*\*Ask the user for the name of the assignment

//ans create this subdirectory if it does not already

//exits

char assignment [INPUTBUF];

printf("\nEnter your assignment (ex: prog1): ");

if(fgets(assignment,INPUTBUF,stdin))

{

assignment[strcspn(assignment,"\n")] = 0;

}

strcat(course,"/");

strcat(course,assignment);

if(verfydirexist(course) == -1)

{

perror (course);

mkdir(course, 0777);

}

strcat(projectpath, course);

//6)Show the user a list of the files in their current

//directory\*\*\*\*\*

int filesnumb = verifyfiles(".");

printf("\nThere are %d files in this directory", filesnumb);

if (filesnumb < 0 )

{

printf ("could not load files in this directory");

exit(0);

}

//7)Ask the user for the names of the files to submit.

//Allow \* for everything

printf("\nPlease enter the name of files you want to submit."

"\n(Put space between files names, or \* for all files)"

"\nSubmit :");

strcat(projectpath,"/");

//check if the files submitted are not null

char submitfiles [INPUTBUF];

if(fgets(submitfiles,INPUTBUF,stdin))

{

submitfiles[strcspn(submitfiles,"\n")] = 0;

}

transferfiles(submitfiles, projectpath);

exit(0);

}

//This function verifies if a directory exists.

int verfydirexist (char \*dirname)

{

struct stat dirstat;

char currentdir [64];

strcpy(currentdir, "./");

strcat(currentdir, dirname);

return stat(currentdir, &dirstat);

}

/\*

\* returns pointer to username associated with uid,

\* uses getpwuid()

\*/

char \*get\_user( uid\_t uid )

{

struct passwd \*pw\_ptr;

static char numstr[10];

if (( pw\_ptr = getpwuid(uid)) == NULL )

{

sprintf(numstr,"%d", uid);

return numstr;

}

else

return pw\_ptr->pw\_name ;

}

/\*displays files in current directory\*/

int verifyfiles (char \*currentdir)

{

int i, numbfiles;

struct dirent \*\*list;

printf("\nBelow are the files in this directory: %s", currentdir);

numbfiles = scandir(currentdir, &list, NULL, alphasort);

if ( numbfiles == -1)

{

perror (currentdir);

}

printf("\n%15s %s %s %3s", "SIZE", "DATE", "TIME", "NAME\n");

for (i = 0; i<numbfiles; i++)

{

dostat( list[i]->d\_name);

}

//after scanning you clear the data in name\_list

for (i = 0; i < numbfiles; i++)

{

free(list[i]);

}

free (list);

return numbfiles - 2; //because we have to substract the hiddenfiles

}

void dostat ( char \*filename)

{

struct stat info;

if (stat(filename, &info) ==-1)

{

perror(filename);

}

else

{

show\_file\_info (filename, &info);

}

}

/\*

\*This function does the display info formatting

\*/

void show\_file\_info (char \*filename, struct stat \*info\_p)

{

if ( S\_ISDIR (info\_p-> st\_mode))

{

return;

}

printf("%15ld", (long) info\_p->st\_size);

printf(" %-13.12s", 4+ctime(&info\_p->st\_mtime));

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

}

/\*

\*This function submits files from src directory to dest directory

\*/

void transferfiles( char \*files, char \*path)

{

int i,j;

i = 0;

char \*submited [512];

//int in\_fd, out\_fd, n\_chars;

char updatepath [512];

//char buf [BUFFERSIZE];

if (strchr(files, ' ') != NULL)

{

submited[i] = strtok(files, " ");

while (submited[i] != NULL)

{

i++;

submited [i] = strtok(NULL, " ");

}

for(j=0; j<i;j++)

{

pathupdated (updatepath,path,submited[j]);

dotransfer (updatepath, submited[j]);

}

verifyfiles(path);

}else if (strchr(files, '\*') != NULL)

{

DIR \*dir\_ptr;

struct dirent \*direntp;

dir\_ptr = opendir(".");

while ((direntp = readdir(dir\_ptr)) != NULL)

{

if(direntp->d\_type !=4)

{

submited[i] = direntp->d\_name;

i++;

}

}

for (j=0; j<i; j++)

{

pathupdated (updatepath,path,submited[j]);

dotransfer (updatepath, submited[j]);

}

verifyfiles(path);

}

}

void pathupdated (char \*updatepath, char \*path, char \*submited)

{

strcpy (updatepath, path);

strcat (updatepath, submited);

}

void dotransfer (char \*newpath, char \*filesub)

{

int in\_fd, out\_fd, n\_chars;

char buf [BUFFERSIZE];

if((in\_fd = open (filesub, O\_RDONLY)) == -1)

{

verifyfiles(newpath);

oops("Cannot open ", filesub);

}

if((out\_fd = creat (newpath, COPYMODE)) == -1)

oops("cannot creat", newpath);

while (( n\_chars = read (in\_fd,buf, BUFFERSIZE)) > 0)

{

if (write( out\_fd, buf, n\_chars) != n\_chars)

oops("Write error to ",newpath);

}

if (n\_chars == -1)

{

oops("read error from", filesub);

}

if (close (in\_fd) == -1 || close (out\_fd) == -1)

{

oops("Error closing files", "");

}

}

void oops(char \*s1, char \*s2)

{

fprintf(stderr,"error: %s ", s1);

perror(s2);

exit(1);

}

COS 350 Program 3 Testing - please fill out and turn in this testing sheet with your written submission. Name(s): Enoch Ikunda

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Pts | Works | Mostly  Works (explain) | Occasionally  Works (explain) | Not implemented | Explanation |
| 1. Ask Course Name & Verify | 5 | yes |  |  |  |  |
| 2. Verify the submit dir exists | 5 | yes |  |  |  |  |
| 3. Get the user’s name | 5 | yes |  |  |  |  |
| 4. Check for and create the user  dir. | 5 | yes |  |  |  |  |
| 5. Ask & create  the assignment dir. | 5 | yes |  |  |  |  |
| 6. Show the user the list of files to  submit. | 25 | yes |  |  |  |  |
| 7. Ask for files to submit. | 5 | yes |  |  |  |  |
| 8.a. Copy the list of files. | 15 | yes |  |  |  |  |
| 8.b. Copy \* | 10 | yes |  |  |  |  |
| 9. Show the final  contents of the submission dir | 5 | yes |  |  |  |  |
| 10. Makefile | 5 | yes |  |  |  |  |
| error handling | 10 | yes |  |  |  | What errors? |