```
// Note: In order for program to run on a remote server you must compile it on the server.
// Ex: From terminal type:
//              ssh YOURNAME@cs360.eecs.wsu.edu
//              enter PWD
//              go to public_html/cgi-bin
//              cc -o mycgi mycgi.c util.o   # generates mycgi executable.
//              chmod u+s mycgi              # chmod mycgi to a SETUID executable
//              Make sure that your public_html webpage directs to the mycgi output and not KCs version
//              You can verify this my runing code. If there is a red background you are running someone
elses code

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <sys/stat.h>
#include <dirent.h>

#define MAX 10000
typedef struct {
        char *name;
        char *value;
} ENTRY;

ENTRY entry[MAX];


myls(char* path, char cwd[]) { // Works
        struct stat stats;
//      struct info *infom;
        DIR *dp = opendir(path);
        if(strlen(path) == 0) {
                strcpy(path, ".");
        }
        //opens a DIR (for R/W), and
        if(dp) {
                struct dirent *ep = readdir(dp);
                //returns ep pointing at the next entry of an opened DIR
                while(ep != NULL){
                        lstat(ep->d_name, &stats);
                        printf("%s \n", ep->d_name);
                        ep = readdir(dp);
                }
        } else {
                printf("Could not open '%s'\n", path);
        }
}

main(int argc, char *argv[])
{
        int i, m;
        char cwd[128];
        char pathName[255];
        FILE* tmp;
        int failed = 0;
        m = getinputs();    // get user inputs name=value into entry[ ]
        getcwd(cwd, 128);   // get CWD pathname

        printf("Content-type: text/html\n\n");
        printf("<p>pid=%d uid=%d cwd=%s\n", getpid(), getuid(), cwd);

        printf("<H1>Echo Your Inputs</H1>");
        printf("You submitted the following name/value pairs:<p>");

        for(i=0; i <= m; i++)
                printf("%s = %s<p>", entry[i].name, entry[i].value);
        printf("<p>");
```

```
        /*************************************************************
         Write YOUR C code here to processs the command
         mkdir dirname
         rmdir dirname
         rm    filename
         cat   filename
         cp    file1 file2
         ls    [dirname] <== ls CWD if no dirname
                ************************************************************/
        //  failed = 0;
        if(strcmp(entry[0].value, "mkdir") == 0){ // mkdir
                for(i = 1;  i <= m; i++) {
                        if(strcmp(entry[i].value, "\0") != 0){ // checks to make sure it's not a NULL,
                                strcpy(pathName, cwd);
                                strcat(pathName, "/");
                                strcat(pathName, entry[i].value);
                                printf("command is %s, Dir name is \"%s\"\n", entry[0].value, pathName);
                                failed = mkdir(pathName, 0777); // was going to use "mode_t mode" from
sys/stat.h (S_IRWXU, S_IRWXG, S_IROTH, S_IXOTH) but friend suggested 0x16D instead and it works
                                if(failed != 0){
                                        i = m; // this ends the loop, in hind sight I could have made a
while function, but this works
                                }
                        }
                }
        }
        else if(strcmp(entry[0].value, "rmdir") == 0){
                for(i = 1;  i <= m; i++) {
                        if(strcmp(entry[i].value, "\0") != 0){ // again, compares to NULL "0\"
                                strcpy(pathName, cwd);
                                strcat(pathName, "/");
                                strcat(pathName, entry[i].value);
                                printf("command is %s, Dir name is \"%s\"\n", entry[0].value, pathName);
                                failed = rmdir(pathName);
                        }
                }
        }
        else if(strcmp(entry[0].value, "rm") == 0){
                for(i = 1;  i <= m; i++){
                        if(strcmp(entry[i].value, "\0") != 0){
                                strcpy(pathName, cwd);
                                strcat(pathName, "/");
                                strcat(pathName, entry[i].value);
                                printf("command is %s, file name is \"%s\"\n", entry[0].value, pathName);
                                printf("Failed to remove\n");
                                failed = 1;
                                failed = remove(pathName);
                        }
                }
        }
        else if(strcmp(entry[0].value, "cat") == 0){
                char buffer[1024] = {'\0'};
                for(i = 1; i <= m; i++){
                        tmp = fopen(entry[i].value, "r");
                        if(tmp == NULL){
                                failed = 1;
                        }
                        else{
                                while(fgets(buffer, 1024, tmp) != NULL){
                                        printf("%s <br/>", buffer);
                                }
                        }
                        if(failed != 0){
                                i = m;
                        }
```

```
                }
                failed = 0;
        }
        else if(strcmp(entry[0].value, "cp") == 0){
                FILE *src, *dest;
                char buffer[1024] = {'\0'};
                src = fopen(entry[1].value, "r");
                dest = fopen(entry[2].value,  "w");
                if(src == NULL || dest == NULL){
                        printf("Could not open file");
                } else {
                        while(fgets(buffer, 1024, src) != NULL){
                                fputs(buffer, dest);
                        }
                }
                failed = 0;
        } else if(strcmp(entry[0].value, "ls") == 0){
                //printf("need to getdents");
                if(strlen(entry[1].value) != 0){
                        myls(entry[1].value, cwd);
                }
                else{
                        myls(".", cwd);
                }
                failed = 0;  // this make it so system knows that command succeeded

        }
        else {
                printf("Command not programmed yet");
        }

        if(failed) {
                printf("<h2>%s failed</h2>", entry[0].value);
                //      printf("Error: %s<br/><br/>", strerror(errno)); //If I want to include error
decoding
        } else {
                printf("<h2>Command succeeded!</h2>");
        }
        failed = 0;


        // create a FORM webpage for user to submit again
        printf("</title>");
        printf("</head>");
        printf("<body bgcolor=\"#c49561\" link=\"#330033\" leftmargin=8 topmargin=8");
        printf("<p>------------------ DO IT AGAIN ----------------\n");

        printf("<FORM METHOD=\"POST\" ACTION=\"http://cs360.eecs.wsu.edu/~juel/cgi-bin/mycgi\">");

        //------ NOTE : CHANGE ACTION to YOUR login name ----------------------------
        //printf("<FORM METHOD=\"POST\" ACTION=\"http://cs560.eecs.wsu.edu/~YOURNAME/cgi-bin/mycgi\">");

        printf("Enter command : <INPUT NAME=\"command\"> <P>");
        printf("Enter filename1: <INPUT NAME=\"filename1\"> <P>");
        printf("Enter filename2: <INPUT NAME=\"filename2\"> <P>");
        printf("Submit command: <INPUT TYPE=\"submit\" VALUE=\"Click to Submit\"><P>");
        printf("</form>");
        printf("-------------------------------------------------<p>");

        printf("</body>");
        printf("</html>");
}
```