```
//Shivam Mevawala
//ECE 357 PS3
//myshell.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <fcntl.h>

int execline(char **argv, char * outname, int mode) {
    pid_t pid,cpid;
    int pos=1;
    struct rusage rtime;
    struct timeval  tstart, tend;
    unsigned status;
    char line[1024];
    line[0]='\0';
    if(sizeof(argv)>1){
        while(argv[pos]!='\0'){
            strcat(line,argv[pos]);
            strcat(line," ");
            pos++;
        }
            argv[pos-1]=NULL;
            line[strlen(line)-2]='\0';
    }
    printf("Executing command %s with argument \"%s\"\n",argv[0], line);
     gettimeofday(&tstart, NULL);
    switch(pid = fork()) {     /* fork a child process*/
        case -1:
            perror("Fork failed");
            exit(1);
            break;
        case 0:
            // printf("In child\n");
            FILE *fp;
            if(mode==0){
                if(execvp(*argv,argv)<0)
                    perror("Execution error");
            }
            else if(mode==1) {
                if((fp=fopen(outname,"r"))==NULL){
                    perror("File opening error");
                    return -1;}
                if(dup2(fileno(fp),0)<0){
                    perror("Dup error");
                    return -1;
                }
                if(execvp(*argv,argv)<0){
                    perror("Execution error");
                    return -1;
                }
                fclose(fp);
            }
            else if(mode==2){
                if((fp=fopen(outname,"a"))==NULL){
                    perror("File opening error");
                    return -1;
                }
                if(dup2(fileno(fp),2)<0){
                    perror("Dup error");
                    return -1;
```

```
            }
            if(execvp(*argv,argv)<0){
                perror("Execution error");
                return -1;
            }
            fclose(fp);
        }
        else if(mode==3){
            if((fp=fopen(outname,"a"))==NULL){
                perror("File opening error");
                return -1;
            }
            if(dup2(fileno(fp),1)<0){
                perror("Dup error");
                return -1;
            }
            if(execvp(*argv,argv)<0){
                perror("Execution error");
                return -1;
            }
            fclose(fp);
        }
        else if(mode==4){
            if((fp=fopen(outname,"w"))==NULL){
                perror("File opening error");
                return -1;
            }
            if(dup2(fileno(fp),2)<0){
                perror("Dup error");
                return -1;
            }
            if(execvp(*argv,argv)<0){
                perror("Execution error");
                return -1;
            }
            if(fclose(fp)==NULL){
                perror("File closing error");
                return -1;
            }
        }
        else if(mode==5){
            if((fp=fopen(outname,"w"))==NULL){
                perror("File opening error");
                return -1;
            }
            if(dup2(fileno(fp),1)<0){
                perror("Dup error");
                return -1;
            }
            if(execvp(*argv,argv)<0){
                perror("Execution error");
                return -1;
            }
            if(fclose(fp)==NULL){
                perror("File closing error");
                return -1;
            }
        }

        break;
    default:
    //back to parent
        while ((cpid=wait(&status)) != pid){
            if (cpid<0){
                perror("Wait failed");
                break;
            }
```

```
            };
             gettimeofday(&tend, NULL);
            if((getrusage(RUSAGE_CHILDREN, &rtime)<0))
                perror("Time error");

            printf("Command returned with return code %d\n",WIFCONTINUED(status));
            printf("consuming %f real seconds, %ld.%.6d user, %ld.%.6d system\n",
                (double) (tend.tv_usec - tstart.tv_usec)/1000000
                + (double) (tend.tv_sec - tstart.tv_sec),rtime.ru_utime.tv_sec,
                rtime.ru_utime.tv_usec,rtime.ru_stime.tv_sec,rtime.ru_stime.tv_usec);
            break;
     }
    return 0;
}

//Parses inputed line into command for execvp
void  parse(char *line, char **argv) {
    while (*line != '\0') {       /* until end of line */
        while (*line == ' ' || *line == '\t' || *line == '\n'){
                    *line='\0';
                    line++;
        }
        *argv++ = line;          /* store the argument position*/
        while (*line != '\0' && *line != ' ' && *line != '\t' && *line != '\n')
            line++;             /* skip the argument until whitespace*/
     }
     *argv = NULL;                 /* mark end of array of array*/
}

//takes input line and determines the mode for file redirection
int main (int argc, char *argv[]) {
    char line[1024], *inargv[128], *first, *second;
    FILE *fp=stdin;
    if(argc>1)
        fp=fopen(argv[1],"r");

    while(fgets (line, 1024, fp)!=NULL){
        if(line[0]!='#'){

            if((second=strstr(line,"<"))!=NULL){
                first=strtok(line,"<");
                parse(first,inargv);
                second=second+2*sizeof(char);
                second[strlen(second)-1]=NULL;
                if(execline(inargv, second, 1))
                    return 1;
                //mode=1
            }

            else if((second=strstr(line,">"))!=NULL){
                first=strtok(line,">");
                parse(first,inargv);

                if(second[1]=='>'){
                    second=second+3*sizeof(char);
                    second[strlen(second)-1]=NULL;

                    if(first[strlen(first)-1]=='2'){
                        if(execline(inargv,second,2))
                            return 1;
                        //mode=2
                    }
                    else{
                        if(execline(inargv,second,3))
                            return 1;
                        //mode=3
                    }
```

```
                }

                else{
                    second=second+2*sizeof(char);
                    second[strlen(second)-1]=NULL;
                    if(first[strlen(first)-1]=='2'){
                        if(execline(inargv,second,4))
                            return 1;
                        //mode=4
                    }
                    else{
                        if(execline(inargv,second,5))
                            return 1;
                        //mode=5
                    }
                }

            }
            else{
                parse(line, inargv);
                if(execline(inargv,"",0))
                    return 1;
            }
        }
    }

    return 0;
}
```