```
/***************************
Programer: Michael Marelli
Class: CS460
Project: Final
Date: 12/9/2016
File: sh.c
***************************/
#include "ucode.c"

main(int argc, char** argv)
{
    char cmd[128], copy[128], *token;
    int pid = 0, status;

    while(1)
    {
        printf("MikeSH#: ");
        bzero(cmd, 128);
        gets(cmd);
        strncpy(copy, cmd, 128);
        token = strtok(copy, " ");
        if(!strcmp("cd", token) || !strcmp("logout", token) || !strcmp("su", token))
        {
            if(!strcmp("cd", token))
            {
                token = strtok(0, ' ');
                chdir(token);
            }
            else if(!strcmp("logout", token))
            {
                exit(1);
            }
            else
            {
                //Need beter protection for su
                printf("Enter passwd for su:");
                gets(cmd);
                if(!strcmp("12345", cmd))
                {
                    chuid(0,0);
                }
            }
        }
        else
        {
            pid = 0;
            pid = fork();
            if(pid) //we are the shell
            {
                if(!strstr(cmd, "&"))
                {
                    pid = wait(&status);
                }
                else
                    continue;
            }
            else
                do_pipe(cmd, 0);
        }
    }
```

```
}

int do_pipe(char *cmdLine, int *pd)
{
	char *head, *tail;
	int lpd[2];
	int hasPipe = 0;

	if(pd) //if has a pipe passed in, as writer on pipe pd:
	{
		close(pd[0]);
		dup2(pd[1], 1);
		close(pd[1]);
	}
	//divide cmdLine into head, tail by rightmost pipe sumbol
	hasPipe = scan(cmdLine, &head, &tail);
	if(hasPipe)
	{
		//create a pipe lpd;
		pipe(lpd);
		pid = 0;
		pid = fork();
		if(pid) //parent
		{
			//as reader on lpd:
			close(lpd[1]);
			dup2(lpd[0], 0);
			close(lpd[0]);
			do_command(tail);
		}
		else
		{
			do_pipe(head, lpd);
		}
	}
	else
	{
		do_command(cmdLine);
	}
}

int do_command(char *cmdLine)
{
	int i = 0;
	char *file;

	while(cmdLine[i])
	{
		if(cmdLine[i] == '<' || cmdLine[i] == '>')
		{
			if(cmdLine[i] == '>') //out redirection working
			{
				cmdLine[i++] = '\0';
				if(cmdLine[i] == '>')
				{
					i++;
					while(cmdLine[i] == ' ')
					{
						i++;
					}
```

```
                    file = cmdLine + i;

                    while(cmdLine[i] != ' ' && cmdLine[i])
                    {
                        i++;
                    }
                    cmdLine[i] = '\0';
                    i++;
                    close(1);
                    open(file, O_WRONLY | O_APPEND | O_CREAT);
                }
                else
                {
                    while(cmdLine[i] == ' ')
                    {
                        i++;
                    }
                    file = cmdLine + i;

                    while(cmdLine[i] != ' ' && cmdLine[i])
                    {
                        i++;
                    }
                    cmdLine[i] = '\0';
                    i++;
                    close(1);
                    open(file, O_WRONLY | O_TRUNC | O_CREAT);
                }
            }
            else
            {
                cmdLine[i++] = '\0';
                while(cmdLine[i] == ' ')
                {
                    i++;
                }
                file = cmdLine + i;

                while(cmdLine[i] != ' ' && cmdLine[i])
                {
                    i++;
                }
                cmdLine[i] = '\0';
                i++;
                close(0);
                open(file, O_RDONLY);
            }
        }
        i++;
    }

    exec(cmdLine);
}

int scan(char* cmdLine, char **head, char **tail)
{
    int i;
    for(i = 128; i > 0; i--)
    {
        if(cmdLine[i] == '|')
```

```
            {
                *tail = cmdLine + i + 1;
                *head = cmdLine;
                cmdLine[i] = '\0';
                return 1;
            }
        }
        *head = cmdLine;
        *tail = 0;
        return 0;
}
```