

```
1 #include<stdio.h>
2 #include<string.h>
3 #include<unistd.h>
4 #include<dirent.h>
5
6 #define MAX_COMMAND_LENGTH 100
7 #define COMMAND_LENGTH 10
8
9 void count(char *option, char *filename){
10     int ch_count=0, word_count =0, line_count=0;
11
12     FILE *fp = fopen(filename, "r");
13     if(fp == NULL){
14         printf("Error opening file.\n");
15         return ;
16     }
17
18     int ch;
19
20     while((ch=fgetc(fp)) != EOF){
21         if(ch == ' ')
22             word_count++;
23         if(ch == '\n'){
24             line_count++;
25             word_count++;
26         }
27         else
28             ch_count++;
29     }
30 }
```



```
30
31 fclose(fp);
32
33 if(strcmp(option, "c")==0){
34     printf("Character count : %d\n", ch_count);
35 }
36 else if(strcmp(option, "w")==0){
37     printf("Word count : %d\n", word_count);
38 }
39 else if(strcmp(option, "l")==0){
40     printf("Line count : %d\n", line_count);
41 }
42 else {
43     printf("Invalid option! Try [c|w|l]\n");
44 }
45
46 }
47
48
49 void list(char *option, char *dirname){
50     DIR *dr;
51
52     dr = opendir(dirname);
53
54     struct dirent *entry;
55
56     if(dr==NULL){
57         printf("Directory not present");
58         return;
59     }
60 }
```



```

55
56 if(dr==NULL){
57     printf("Directory not present");
58     return;
59 }
60
61
62 if(strcmp(option, "f")==0){
63     while( (entry = readdir(dr)) != NULL){
64         printf("\t%s\n", entry->d_name);
65     }
66 }
67 else if(strcmp(option, "n") == 0){
68     int n=0;
69     while( (entry = readdir(dr)) != NULL){
70         n++;
71     }
72     printf("No of entries : %d\n", n);
73 }
74 else if(strcmp(option, "i") == 0){
75     while( (entry = readdir(dr)) != NULL){
76         printf("\t%s\t%d\n", entry->d_name, entry->d_ino );
77     }
78 }
79
80
81
82 }
83 void typeline(char *option, char *filename, char *no_of_lines){
84     FILE *fp;

```



```
80
81
82 }
83 void typeline(char *option, char *filename, char *no_of_lines){
84     FILE *fp;
85     fp = fopen(filename, "r");
86
87     if(fp == NULL){
88         printf("Error opening file\n");
89         return;
90     }
91
92     int ch;
93
94     if(strcmp(option, "n")==0){
95         int n = atoi(no_of_lines);
96         int i=0 ;
97         printf("\n");
98         while((ch = fgetc(fp)) != EOF){
99             if (n == i){
100                 break;
101             }
102             printf("%c", ch);
103
104             if(ch == '\n')
105                 i++;
106         }
107         printf("\n");
108     }
109     else if(strcmp(option, "-n")==0){
```



```
109     else if(strcmp(option, "-n")==0){
110         int n = atoi(no_of_lines);
111         int totalLines = 0;
112         int currentLine = 0;
113
114         while((ch=fgetc(fp)) != EOF){
115             if(ch == '\n')
116                 totalLines++;
117         }
118
119         int lines_to_skip = totalLines - n;
120         //printf("total lines and lines to skip are : %d %d", totalLines, lines_to_skip);
121
122         fseek(fp, 0, 0);
123
124         while((ch=fgetc(fp)) != EOF){
125             //printf("\t%c", ch);
126             if(ch == '\n')
127                 currentLine++;
128
129             if(currentLine < lines_to_skip){
130                 continue;
131             }
132             else {
133                 printf("%c", ch);
134             }
135         }
136
```



```
138
139
140     }
141     printf("\n");
142
143 }
144
145 else if(strcmp(option, "a") == 0){
146     printf("\n");
147     while((ch = fgetc(fp)) != EOF){
148         printf("%c", ch);
149     }
150     printf("\n");
151 }
152
153 }
154
155 void main() {
156
157     char line[MAX_COMMAND_LENGTH];
158     char *commands[COMMAND_LENGTH];
159
160     while(1) {
161         printf("myshells ");
162         fgets(line, MAX_COMMAND_LENGTH, stdin);
163
164         if(strcmp(line, "exit")==0){
165             break;
166         }
167     }
```



I



A1.c X ex.txt X

```
163
164     if(strcmp(line, "exit")==0){
165         break;
166     }
167
168     int comm_count=0;
169
170     char *token = strtok(line, " \n");
171
172     while(token != NULL){
173         commands[comm_count] = token;
174         token = strtok(NULL, " \n");
175         comm_count++;
176     }
177     commands[comm_count] = NULL;
178
179     if(strcmp(commands[0], "count") == 0){
180         // count function
181         count(commands[1], commands[2]);
182     }
183     else if(strcmp(commands[0], "list") == 0){
184         list(commands[1], commands[2]);
185     }
186     else if(strcmp(commands[0], "typeline") == 0){
187         typeline(commands[1], commands[2], commands[3]);
188     }
189     else {
190         // Create a child process.
191         if(fork()==0){
192             char path[COMMAND_LENGTH] = "/bin/";
```


File Edit View Search Tools Documents Help

  Open  Save   Undo     A1.c  ex.txt 

```
177     commands[comm_count] = NULL;
178
179     if(strcmp(commands[0], "count") == 0){
180         // count function
181         count(commands[1], commands[2]);
182     }
183     else if(strcmp(commands[0], "list") == 0){
184         list(commands[1], commands[2]);
185     }
186     else if(strcmp(commands[0], "typeline") == 0){
187         typeline(commands[1], commands[2], commands[3]);
188     }
189     else {
190         // Create a child process.
191         if(fork()==0){
192             char path[COMMAND_LENGTH] = "/bin/";
193             strcat(path, commands[0]);
194             //printf("path : %s\n", path);
195
196             execv(path, commands);
197         }
198         else {
199             int status;
200             wait(&status);
201         }
202     }
203
204
205 }
206 }
```