

Branch: master ▾

Find file

Copy path

[cs-35l](#) / [assignment4](#) / [sfrob.c](#)

prithvikannan added to lab4 and fixed comment is sfrob

ffd6308 on Oct 28

[1 contributor](#)

Raw

Blame

History



170 lines (150 sloc) 4.21 KB

```
1  #include <stdbool.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  // implements comparison between a and b without deobfuscating
6  int frobcmp(char const *a, char const *b)
7  {
8      // make sure pointers are not null
9      if (a != 0 && b != 0)
10     {
11         // iterate through char array with pointers a and b
12         while (*a != ' ' && *b != ' ')
13         {
14             // unfrobnicate a single byte
15             char a_i = *a ^ 42;
16             char b_i = *b ^ 42;
17
18             // compare a and b and check which ends first
19             if (a_i < b_i || *a == ' ')
20             {
21                 return -1;
22             }
23             else if (a_i > b_i || *b == ' ')
24             {
25                 return 1;
26             }
27
28             a++;
29             b++;
30         }
31     }
32     // a and b always equal
33     return 0;
34 }
35
36 // custom comparator that calls frobcmp
37 int cmp(const void *a, const void *b)
38 {
39     return frobcmp(*(char **)a, *(char **)b);
40 }
41
42 int main()
43 {
44     char current_char;
45
46     // allocate memory for array of strings
47     char **arr = (char **)malloc(sizeof(char *));
48     // allocate memory for new string
49     char *temp_string = (char *)malloc(sizeof(char));
50     if (arr == NULL || temp_string == NULL)
51     {
```

```
52     fprintf(stderr, "Memory allocation error");
53     exit(1);
54 }
55
56 int string_ptr = 0;
57 int char_ptr = 0;
58 arr[0] = temp_string;
59 bool needNewString = false;
60
61 // increases size of string array to hold one more string
62 arr = (char **)realloc(arr, (string_ptr + 1) * sizeof(char *));
63 // creates the first string
64 temp_string = (char *)malloc(sizeof(char));
65 if (arr == NULL || temp_string == NULL)
66 {
67     fprintf(stderr, "Memory allocation error");
68     exit(1);
69 }
70
71 // read from stdin until eof or error
72 while (true)
73 {
74     // read text from standard input
75     current_char = getchar();
76     if (ferror(stdin))
77     {
78         fprintf(stderr, "Input read error");
79         exit(1);
80     }
81     else if (feof(stdin))
82     {
83         // hit end of file, exit while loop
84         break;
85     }
86
87     if (!needNewString)
88     {
89         temp_string = (char *)realloc(temp_string, (char_ptr + 1) * sizeof(char));
90         if (temp_string == NULL)
91         {
92             fprintf(stderr, "Memory allocation error");
93             exit(1);
94         }
95
96         // space is delimiter of new strings
97         if (current_char == ' ')
98         {
99             needNewString = true;
100         }
101     }
102     else // if program must create a new string
103     {
104         char_ptr = 0;
105
106         // handle consecutive spaces by skipping iteration
107         if (current_char == ' ' && char_ptr == 0)
108         {
109             continue;
110         };
111
112         string_ptr++;
113         needNewString = false;
114
115         arr = (char **)realloc(arr, (string_ptr + 1) * sizeof(char *));
116         temp_string = (char *)malloc(sizeof(char));
117         if (arr == NULL || temp_string == NULL)
```

```
118     {
119         fprintf(stderr, "Memory allocation error");
120         exit(1);
121     }
122
123 }
124
125 // add new char after adjusting pointers and allocating memory
126 temp_string[char_ptr] = current_char;
127 arr[string_ptr] = temp_string;
128 char_ptr++;
129 }
130
131 // add trailing space if not present
132 if (string_ptr != -1 && arr[string_ptr][char_ptr - 1] != ' ')
133 {
134     temp_string = (char *)realloc(temp_string, (char_ptr + 1) * sizeof(char));
135     if (temp_string == NULL)
136     {
137         fprintf(stderr, "Memory allocation error");
138         exit(1);
139     }
140     temp_string[char_ptr] = ' ';
141     arr[string_ptr] = temp_string;
142 }
143
144 qsort(arr, string_ptr + 1, sizeof(char *), cmp);
145
146 // print to stdout
147 int i;
148 for (i = 0; i < string_ptr + 1; i++)
149 {
150     int j = 0;
151     while (true)
152     {
153         if (putchar(arr[i][j]) == EOF)
154         {
155             fprintf(stderr, "Printing error");
156             exit(1);
157         }
158
159         // if space then move to next line
160         if (arr[i][j] == ' ')
161         {
162             break;
163         }
164         j++;
165     }
166     free(arr[i]);
167 }
168 free(arr);
169 exit(0);
170 }
```