

Branch: master ▾

Find file

Copy path

[cs-35l](#) / [assignment5](#) / [report.txt](#)

prithvikannan added report

5053b6c on Nov 5

[1 contributor](#)

Raw

Blame

History



217 lines (185 sloc) 8.99 KB

```

1 1. wrote tr2b.c
2 I learned that in C the input count is passed in as an int argc and the inputs
3 are passed in as a char pointer argv. First I checked the input values to make
4 sure they were correct by checking if exactly 3 operands were given and that the
5 from and to string were the same length.
6
7 Then I verified that there were no duplicate letters, and built my dictionary
8 between the letters to translate.
9
10 Next I used getChar() to read a character, and if I had a translation for that
11 letter, I would apply it and use putChar(). Otherwise, I would just use
12 putChar() and go to the next character.
13
14 To compile, I used: gcc -std=c11 tr2b.c -o tr2b
15
16 2. wrote tr2u.c
17 Similar to tr2b.c, I started with input checking and then moved to creating my
18 dictionary.
19
20 However, instead of using getChar() and putChar(), I used a temporary buffer of
21 size 1 and the read() command to take in a single character. Then I would check
22 if that character could be translated and translate if necessary and then output
23 using the write() command.
24
25 To compile, I used: gcc -std=c11 tr2u.c -o tr2u
26
27 3. testing
28
29 man head
30     Looked at the documentation for head, which takes the first x bytes of a
31     file when used with the --bytes flag.
32 head --bytes=5000000 /dev/urandom > tester.txt
33     ran this script to create a random file of 5000000 bytes.
34
35 man strace
36     Looked at the documentation of strace, realized that I needed to pass
37     -c flag for easy counting.
38
39 strace -c ./tr2b 'A' 'B' < tester.txt > result_b.txt
40 strace -c ./tr2u 'A' 'B' < tester.txt > result_u.txt
41     Ran strace on buffered and unbuffered tr commands on my test input file.
42     I set from to 'A' and to to 'B' arbitrarily, and piped the outputs to files.
43
44 tr2b:
45 % time      seconds  usecs/call   calls   errors syscall
46 -----
47 0.00      0.000000         0        2         read
48 0.00      0.000000         0        1         write
49 0.00      0.000000         0        2         open
50 0.00      0.000000         0        2         close
51 0.00      0.000000         0        4         fstat

```

```

52  0.00  0.000000      0    10      mmap
53  0.00  0.000000      0      3      mprotect
54  0.00  0.000000      0      1      munmap
55  0.00  0.000000      0      1      brk
56  0.00  0.000000      0      1      1 access
57  0.00  0.000000      0      1      execve
58  0.00  0.000000      0      1      arch_prctl
59  -----
60  100.00  0.000000          29      1 total

```

```

63  tr2u:
64  % time    seconds  usecs/call    calls    errors syscall
65  -----
66  56.25    0.484831      0    5000000      write
67  43.75    0.377132      0    5000002      read
68  0.00     0.000000      0      2      open
69  0.00     0.000000      0      2      close
70  0.00     0.000000      0      2      fstat
71  0.00     0.000000      0      8      mmap
72  0.00     0.000000      0      3      mprotect
73  0.00     0.000000      0      1      munmap
74  0.00     0.000000      0      1      brk
75  0.00     0.000000      0      1      1 access
76  0.00     0.000000      0      1      execve
77  0.00     0.000000      0      1      arch_prctl
78  -----
79  100.00    0.861963          1000024      1 total

```

```

82  strace -c ./tr2b 'A' 'B' < tester.txt
83  strace -c ./tr2u 'A' 'B' < tester.txt
84      Ran strace on buffered and unbuffered tr commands on my test input file.
85      I set from to 'A' and to to 'B' arbitrarily, and had it output to the
86      terminal.
87
88      I got 29 system calls for tr2b and 10000024 for tr2u, just as above.

```

4. timing the runs

```

93  time ./tr2b 'A' 'B' < tester.txt > result_b.txt
94  time ./tr2u 'A' 'B' < tester.txt > result_u.txt
95      Use the time command to keep track of how long the process took to run tr
96      buffered and unbuffered.

```

```

98  tr2b:
99  real    0m0.004s
100 user    0m0.000s
101 sys     0m0.002s

```

```

103 tr2u:
104 real    0m9.232s
105 user    0m1.377s
106 sys     0m7.813s

```

ANALYSIS OF SFROB AND SFROBU

I created test files of various sizes using these commands

```

112 head --bytes=0 /dev/urandom > zero.txt
113 head --bytes=100 /dev/urandom > hundred.txt
114 head --bytes=10000 /dev/urandom > tenthousand.txt

```

I tested performance of sfrobu.

For sfrobu, I would estimate $(0.03 - 0.003) / (10000 - 100) \times 0.003$

```

118     time ./sfrobu < zero.txt
119         real    0m0.003s
120         user    0m0.001s
121         sys     0m0.002s
122     time ./sfrobu < hundred.txt
123         real    0m0.003s
124         user    0m0.002s
125         sys     0m0.002s
126     time ./sfrobu < tenthousand.txt
127         real    0m0.030s
128         user    0m0.012s
129         sys     0m0.016s
130
131 I tested performance of sfrob.
132 For sfrob, I would estimate  $(0.004 - 0.003) / (10000 - 100) \times 0.003$ 
133     time ./sfrob < zero.txt
134         real    0m0.003s
135         user    0m0.001s
136         sys     0m0.002s
137     time ./sfrob < hundred.txt
138         real    0m0.003s
139         user    0m0.003s
140         sys     0m0.001s
141     time ./sfrob < tenthousand.txt
142         real    0m0.004s
143         user    0m0.000s
144         sys     0m0.003s
145
146 The big O runtime of quicksort is  $O(n \log n)$ , so our function is going to have
147 similar runtime.
148
149 I ran strace to look at the system calls for different sized inputs to my sfrobu
150
151 strace -c ./sfrobu < zero.txt
152      % time   seconds  usecs/call   calls   errors syscall
153      -----
154      20.00    0.000010          3         3      mprotect
155      16.00    0.000008          2         4       fstat
156      16.00    0.000008          8         1      munmap
157      14.00    0.000007          2         4       brk
158      12.00    0.000006          2         3       read
159      12.00    0.000006          1         7      mmap
160      6.00     0.000003          3         1  arch_prctl
161      4.00     0.000002          1         2      close
162      0.00     0.000000          0         2      open
163      0.00     0.000000          0         1    1 access
164      0.00     0.000000          0         1     execve
165      -----
166      100.00    0.000050          29         1 total
167 strace -c ./sfrobu < hundred.txt
168      % time   seconds  usecs/call   calls   errors syscall
169      -----
170      0.00     0.000000          0          3       read
171      0.00     0.000000          0       101      write
172      0.00     0.000000          0          2      open
173      0.00     0.000000          0          2      close
174      0.00     0.000000          0          4       fstat
175      0.00     0.000000          0          7      mmap
176      0.00     0.000000          0          3  mprotect
177      0.00     0.000000          0          1      munmap
178      0.00     0.000000          0          4       brk
179      0.00     0.000000          0          1    1 access
180      0.00     0.000000          0          1     execve
181      0.00     0.000000          0          1  arch_prctl
182      -----
183      100.00    0.000000       130         1 total

```

```

184 strace -c ./sfrobu < tenthousand.txt
185 % time      seconds  usecs/call   calls   errors syscall
186 -----
187 99.96    0.037056         4    10000         write
188  0.04    0.000013         2         8         brk
189  0.00    0.000000         0         3         read
190  0.00    0.000000         0         2         open
191  0.00    0.000000         0         2         close
192  0.00    0.000000         0         4         fstat
193  0.00    0.000000         0         7         mmap
194  0.00    0.000000         0         3         mprotect
195  0.00    0.000000         0         1         munmap
196  0.00    0.000000         0         1    1 access
197  0.00    0.000000         0         1         execve
198  0.00    0.000000         0         1         arch_prctl
199 -----
200 100.00    0.037069        10033         1 total
201
202 The system calls for memory allocations are found by looking at the man page
203 for each of the syscalls from the stack trace.
204
205 man brk
206     brk - change data segment size
207     int brk(void *addr);
208 man mmap
209     mmap - map or unmap files or devices into memory
210     void *mmap(void *addr, size_t length, int prot, int flags,
211               int fd, off_t offset);
212 man munmap
213     munmap - map or unmap files or devices into memory
214     int munmap(void *addr, size_t length);
215
216 So brk is the equivalent of realloc, mmap is the equivalent of malloc, and
217 munmap is the equivalent of free.

```