Kareem Abdelaty

30075331

CPSC 457

Assignment 1

Q1

a –

```
time python3 palindrome.py < t4.txt          time ./slow-pali < t4.txt
Longest palindrome: redder                   Longest palindrome: redder


real    0m0.361s                             real    0m2.873s
user    0m0.317s                             user    0m1.402s
sys     0m0.006s                             sys     0m1.457s


time python3 palindrome.py < t3.txt          time ./slow-pali < t3.txt
Longest palindrome: ___o.O.o___              Longest palindrome: ___o.O.o___


real    0m0.028s                             real    0m0.011s
user    0m0.020s                             user    0m0.005s
sys     0m0.003s                             sys     0m0.000s
```

b-

The C++ code spent 0.000s running t3.txt and 1.402s running t4.txt in kernel mode compare to the python code spending 0.003s running t3.txt and 0.006s running t4.txt in kernel mode

c-

Usually C++ code is faster than python code because C++ is a compiled language while python is interpreted language which means there is a lot of overhead is needed to run the most basic python code. In this case, python is faster on longer files because the C++ code invokes a system call for every byte read which is very inefficient and causes a lot of time to be spent inside kernel mode in addition to the time spent context switching between user and kernel modes.

Q3- Fast-pali performance

```
time ./fast-pali < t3.txt
Longest palindrome: ___o.O.o___
real    0m0.011s
user    0m0.003s
sys     0m0.003s
```

```
time ./fast-pali < t4.txt
Longest palindrome: redder
real    0m0.109s
user    0m0.093s
sys     0m0.004s
```

strace for fast-pali

```
strace -c time ./fast-pali < t4.txt
Longest palindrome: redder
0.08user 0.00system 0:00.08elapsed 97%CPU (0avgtext+0avgdata 4020maxresident)k
0inputs+0outputs (0major+382minor)pagefaults 0swaps
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 76.95    0.003032        3032         1           wait4
  9.37    0.000369           3       112           write
  4.39    0.000173         173         1           execve
  2.34    0.000092           5        18        16 openat
  1.52    0.000060           8         7           mmap
  1.19    0.000047          47         1           clone
  0.91    0.000036           4         8         7 stat
  0.69    0.000027           6         4           mprotect
  0.63    0.000025           6         4           rt_sigaction
  0.53    0.000021          21         1           munmap
  0.38    0.000015           3         4           read
  0.25    0.000010           3         3           lseek
  0.20    0.000008           4         2         1 arch_prctl
  0.18    0.000007           3         2           close
  0.18    0.000007           3         2           fstat
  0.18    0.000007           7         1         1 access
  0.10    0.000004           4         1           brk
------ ----------- ----------- --------- --------- ----------------
100.00    0.003940          22       172        25 total
```

Strace for slow-pali

```
strace -c time ./slow-pali < t4.txt
Longest palindrome: redder
1.43user 1.45system 0:02.88elapsed 99%CPU (0avgtext+0avgdata 3048maxresident)k
48inputs+0outputs (0major+125minor)pagefaults 0swaps
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 99.96    1.451900     1451900         1           wait4
  0.03    0.000445           3       112           write
  0.00    0.000066          16         4           mprotect
  0.00    0.000034          34         1           clone
  0.00    0.000026           6         4           rt_sigaction
  0.00    0.000024           3         7           mmap
  0.00    0.000010           2         4           read
  0.00    0.000008           8         1           munmap
  0.00    0.000007           2         3           lseek
  0.00    0.000004           0        18        16 openat
  0.00    0.000003           1         2           close
  0.00    0.000003           1         2         1 arch_prctl
  0.00    0.000002           1         2           fstat
  0.00    0.000000           0         8         7 stat
  0.00    0.000000           0         1           brk
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
------ ----------- ----------- --------- --------- ----------------
100.00    1.452532        8444       172        25 total
```

a- By comparing the time it took fast-pali to process t3.txt and t4.txt, we can see that fast-pali is slightly faster than slow-pali on the small t3.txt file and much faster than slow-pali on the large t4.txt. Fast-pali is faster than slow-pali because fast pali preforms far less system calls compared to slow-pali. As we can by the strace output above fast-pali makes 22 system calls compared to the 8444 system calls done by slow-pali.

b- By comparing the time it took fast-pali to process t3.txt and t4.txt, we can see that fast-pali is slightly faster than palindrome.py on the small t3.txt file and much faster than it on the large t4.txt. fast-pali is simply faster due to interpreter overhead needed to run the python code.