Operating Systems
Assignment 1

Fezile Manana

September 27, 2018

Problem 1.1

a) Find source code attached.

b) Measurements were taken for both system($scat-s$) and library ($scat-l$) calls:

Library calls $/s$		
real time	user time	sys time
0.007	0.006	0.001
0.007	0.006	0.001
0.007	0.007	0.000
0.007	0.007	0.000
0.007	0.004	0.003
0.008	0.004	0.003
0.007	0.006	0.000
0.008	0.004	0.003
0.007	0.004	0.003
0.008	0.004	0.003

System Call $/s$		
real time	user time	sys time
0.173	0.051	0.121
0.151	0.049	0.101
0.174	0.041	0.132
0.149	0.036	0.112
0.176	0.040	0.135
0.171	0.056	0.115
0.176	0.041	0.133
0.149	0.056	0.091
0.172	0.070	0.102
0.167	0.029	0.138

From the tables it's clear that the library calls spend more time in user space than in the system kernel, whereas the system call variation of the program spends the majority of it's run time in the system kernel. In addition, the system call variation has a longer overall runtime.
This is because the *getc*() and *putc*() library calls put the data to be written to the standard output in a buffer before requesting services from the kernel to print. This means that the *getc*()/*putc*() loop only goes into the kernel once to print everything stored in the buffer. This is the reason the library call variation spends most of it's run time in user space.
On the other hand the *read*()/*write*() system calls invoke the kernel everytime the functions are called. This results in reduced efficiency as

every system call needs to go from user space into kernel space and back into user space. This is an expensive operation, and this is why the system call version of the progam has a longer runtime, and is is also the reason most of the runtime is spent in system mode.
This is confirmed after investigating *ltrace* and *strace*. For the library function calls only one *read*() and one *write*(), whereas for the *read*()/*write*() loop there are as many calls as there are characters.

Problem 1.2

Find source code attached.