Lab 1C Report – Michael Zhang, 404606017

Benchmark test cases

Test case 1
```
./simpsh --profile --rdonly a.txt  --creat  --wronly test1output.txt -
-creat --wronly test1err.txt --pipe --pipe --command 0 4 2 cat a.txt
a.txt --command 3 6 2 grep h --command 5 1 2 sort --wait

cat a.txt a.txt | grep h | sort > test1output.txt 2> test1err.txt
```

Where a.txt was a relatively small file filled with text

Test case 2
```
./simpsh --profile --rdonly a0.txt --creat --wronly test2output.txt --
creat --wronly test2err.txt --pipe --pipe --pipe --profile --command 0
4 2 cat --command 3 6 2 sed s/the/hello/g --command 5 8 2 sed/a/A/g --
command 7 1 2 sort

cat a0.txt | sed s/the/hello/g | sed s/a/A/g  | sort >test2output.txt
2>test2err.txt
```

In this case, a0.txt was the large file provided by Zhaoxing

Test case 3
```
./simpsh --profile --rdonly a2.txt --creat --wronly test3output.txt --
creat --wronly test3err.txt --pipe --pipe --pipe --profile --command 0
4 2 cat --command 3 6 2 sed s/the/hello/g --command 5 8 2 sed/a/A/g --
command 7 1 2 sort –wait

cat a2.txt | sed s/the/hello/g | sed s/a/A/g  | sort >test3output.txt
2>test3err.txt
```

Where a2.txt was a moderately sized file. This was very similar to test case 2, except using a smaller file and the –wait modifier for simpsh.

Benchmark values

Test case	Simpsh	Bash	dash
1			
User time	.0027	.0021	0
System time	.0037	.0033	0
2			
User time	.0045	9.267	9.275
System time	.0063	.0615	.58
3			
User time	.0062	.005	0
System time	.0076	.0072	0

Conclusions
It seems in general that dash is the most efficient way of running commands, followed by bash, and then my simpsh implementation. Dash was so quick/didn't have enough precision so that the value was often zero. This seems to make sense…the system time for each implementation is similar, but the user time is often a little higher for simpsh. This is probably because the simpsh implementation has to run some amount of user facing C code before it can call system calls. Additionally, dash has less overhead then bash when running scripts.

However, it is important to notice that in test case 2 that simpsh was **significantly** faster than the two other implementations. Why is this? Test case 2 was the only implementation that did not call the –wait flag. When dealing with a 73MB file, this can cause a significant difference. It's probable to assume that the bash and dash implementations waited for cat, sed, sed, and sort to complete one by one, while my implementation didn't wait for each command to finish before passing it on to the next. While this can be dangerous in some cases – the resulting output files matched – with a much less user time. This outlier still makes sense though, as the system times remain relatively similar, meaning that the main difference between the implementations is user time.

So, in general the most efficient shell is dash, followed by bash and simpsh. However, in cases while using pipes where waiting for the previous command is not important, simpsh can represent a much faster solution.