Nathan Kong

204 401 093

COM SCI 111 – 1A

Lab 1C: Simple Shell

The commands that I used to test between **simpsh**, **bash**, and **dash** are:

****bash & dash can use the same shell script, simply use ‘./bm.sh’ and ‘dash bm.sh’**

****Tests were conducted on my own machine, I noticed there is a significant difference in comparison to lnxsrv09 (about double).**

[test #1]

simpsh:

```
./simpsh --verbose --profile \
--rdonly a0.txt \
--pipe \
--creat --trunc --wronly a1.txt \
--creat --trunc --wronly a2.txt \
--command 0 2 4 tr A-Z a-z \
--command 1 3 4 sort \
--wait
```

bash/dash:

```
touch a1.txt && touch a2.txt && cat a0.txt | tr A-Z a-z | sort > a1.txt
times
```

[test #2]

simpsh:

```
./simpsh --profile --verbose \
--rdonly a0.txt \
--pipe \
--pipe \
--creat --trunc --wronly a1.txt \
--creat --trunc --wronly a2.txt \
--command 0 2 6 tr A-Z a-z \
--command 1 4 6 sort \
--command 3 5 6 grep you \
--wait
```

bash/dash:

```
touch a1.txt && touch a2.txt && cat a0.txt | tr A-Z a-z | sort | grep you > a1.txt
```

times

simpsh:

```
./simpsh --profile --verbose \
--rdonly a0.txt \
--pipe \
--pipe \
--creat --trunc --wronly a1.txt \
--creat --trunc --wronly a2.txt \
--ignore 2 \
--command 0 2 6 tr A-Z a-z \
--command 1 4 6 sort \
--command 3 5 6 kill -2 $$ \
--wait
```

bash/dash:

```
trap ‘’ INT
cat a0.txt | tr A-Z a-z | sort | kill -2 $$
times
```

	test #1	test #2	test #3
simpsh	user: 2.400 s sys: 0.219 s	user: 2.485 s sys: 0.193 s	user: 2.296 s sys: 0.184 s
bash	user: 2.441 s sys: 0.257 s	user: 2.476 s sys: 0.248 s	user: 2.312 s sys: 0.200 s
dash	user: 2.573 s sys: 0.250 s	user: 2.55 s sys: 0.250 s	user: 2.217 s sys: 0.177 s

Conclusions:

Based from the information gained from the benchmark tests between the three shells, there is no one single "winner". In the basic tests, **simpsh** could beat the others in spending less time on both the user and system. However, when we add **kill** commands and signal handlers (which ignore the signal) to the mix we start to see a better performance in **dash**. The simple shell we have made through this lab is efficient in being able to fork multiple processes and run them, whereas **dash** may have a different implementation which better utilizes each command for speed. We notice that **bash** does not "win" under any of these three tests, and in more extensive comparisons between various shells it is known that **bash** tends to be the slowest of all but is renowned for its robustness and functionality.