Bradley Mont
CS 111, Prof. Eggert
Lab 1A, Tiard
UID: 804-993-030

Lab 1C: Comparing Performance of simpsh, bash, and dash

Benchmark #1

POSIX Shell Command:

```
(tr a-z A-Z < pg98_100.txt | tr a b | sort -f >> append.txt) 2>
err.txt
```

simpsh Implementation:

```
./simpsh \
--rdonly pg98_100.txt \
--creat --append --wronly append.txt \
--creat --trunc --wronly err.txt \
--pipe \
--pipe \
--command 0 4 2 tr a-z A-Z \
--command 3 6 2 tr a b \
--command 5 1 2 sort -f \
--close 4 \
--close 6 \
--wait
```

User and System CPU Time:

For simpsh, I calculated the times by calling getrusage(2) at the end of my program. This function returns a struct with information about the program's user CPU time and system CPU time. I called the function with RUSAGE_SELF and RUSAGE_CHILDREN, and I added the relevant times.

For bash, I created a bash script called bench1bash.sh with my POSIX shell implementation in it (and #!/bin/bash at the top). I then executed the following commands:

```
$ ./bench1bash.sh
$ times
```

For dash, I created a dash script called bench1dash.sh with my POSIX shell implementation in it (and #!/bin/sh at the top). I then executed the following commands:

```
$ ./bench1dash.sh
$ times
```

The times command then outputted information about the previous program's user CPU time and system CPU time in the form:

user time	system time
user time of children	system time of children

bash			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	1.936s	0.582s	2.518s
Trial 2	1.937s	0.607s	2.544s
Trial 3	1.884s	0.638s	2.522s
Average	1.919s	0.609s	2.528s

dash			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	1.949s	0.625s	2.574s
Trial 2	1.957s	0.621s	2.578s
Trial 3	1.879s	0.614s	2.493s
Average	1.928s	0.620s	2.548s

simpsh			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	1.766350s	0.494405s	2.260755s
Trial 2	1.882441s	0.531356s	2.413797s
Trial 3	1.758008s	0.559360s	2.317368s
Average	1.802266s	0.528374s	2.330640s

Benchmark #2

POSIX shell command:

```
(cat pg98_100.txt | egrep -o "r.*t" | sort -r | wc -w > out.txt)
2> err2.txt
```

simpsh implementation:

```
./simpsh \
--rdonly pg98_100.txt \
--creat --wronly out.txt \
--pipe \
--pipe \
--pipe \
--creat --wronly err2.txt \
--command 0 3 8 cat \
--command 2 5 8 egrep -o "r.*t" \
--command 4 7 8 sort -r \
--command 6 1 8 wc -w \
--close 3 \
--close 5 \
--close 7 \
--wait
```

User and System CPU Time:

For simpsh, I calculated the times by calling getrusage(2) at the end of my program. This function returns a struct with information about the program's user CPU time and system CPU time. I called the function with RUSAGE_SELF and RUSAGE_CHILDREN, and I added the relevant times.

For bash, I created a bash script called bench2bash.sh with my POSIX shell implementation in it (and #!/bin/bash at the top). I then executed the following commands:

```
$ ./bench2bash.sh
$ times
```

For dash, I created a dash script called bench2dash.sh with my POSIX shell implementation in it (and #!/bin/sh at the top). I then executed the following commands:

```
$ ./bench2dash.sh
$ times
```

The times command then outputted information about the previous program's user CPU time and system CPU time in the form:

```
user time                 system time
user time of children     system time of children
```

bash			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	2.430s	0.315s	2.745s
Trial 2	2.410s	0.338s	2.748s
Trial 3	2.446s	0.326s	2.772s
Average	2.429s	0.326s	2.755s

dash			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	2.467s	0.331s	2.798s
Trial 2	2.390s	0.380s	2.770s
Trial 3	2.435s	0.340s	2.775s
Average	2.431s	0.350s	2.781s

simpsh			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	2.516921s	0.271490s	2.788411s
Trial 2	2.352554s	0.271379s	2.623933s
Trial 3	2.428007s	0.256134s	2.684141s
Average	2.432494s	0.266334s	2.698828s

Benchmark #3

POSIX shell command:

```
(sed 's/his/her/' < pg98_100.txt | tr abc xyz | sort -d | egrep
-o "n.*s" > out3.txt) 2> err3.txt
```

simpsh implementation:

```
./simpsh \
--rdonly pg98_100.txt \
--creat --wronly out3.txt \
--pipe \
--pipe \
--pipe \
--creat --wronly err3.txt \
--command 0 3 8 sed 's/his/her/' \
--command 2 5 8 tr abc xyz \
--command 4 7 8 sort -d \
--command 6 1 8 egrep -o "n.*s" \
--close 3 \
--close 5 \
--close 7 \
--wait
```

User and System CPU Time:

For simpsh, I calculated the times by calling getrusage(2) at the end of my program. This function returns a struct with information about the program's user CPU time and system CPU time. I called the function with RUSAGE_SELF and RUSAGE_CHILDREN, and I added the relevant times.

For bash, I created a bash script called bench3bash.sh with my POSIX shell implementation in it (and #!/bin/bash at the top). I then executed the following commands:

```
$ ./bench3bash.sh
$ times
```

For dash, I created a dash script called bench3dash.sh with my POSIX shell implementation in it (and #!/bin/sh at the top). I then executed the following commands:

```
$ ./bench3dash.sh
$ times
```

The times command then outputted information about the previous program's user CPU time and system CPU time in the form:

```
user time                system time
user time of children    system time of children
```

bash			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	4.805s	0.709s	5.514s
Trial 2	4.724s	0.697s	5.421s
Trial 3	4.554s	0.688s	5.242s
Average	4.694s	0.698s	5.392s

dash			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	4.751s	0.682s	5.433s
Trial 2	4.820s	0.708s	5.528s
Trial 3	4.717s	0.633s	5.350s
Average	4.763s	0.674s	5.437s

simpsh			
	User CPU Time	System CPU Time	Total CPU Time
Trial 1	4.723878s	0.612285s	5.336163s
Trial 2	4.496698s	0.630358s	5.127056s
Trial 3	4.583764s	0.572894s	5.156658s
Average	4.601447s	0.605179s	5.206626s

Conclusion

For all three of my benchmarks, the simpsh version took up the least CPU time, and the dash version took up the most CPU time. Not to mention, the bash and dash versions took up extremely similar amounts of CPU time, but on average, dash took up the most time for all benchmarks. Based on **this data only**, we can conclude that **for my simpsh implementation and test cases**, the simpsh version was the most efficient, and the dash version was the least efficient in terms of CPU time used. The above results are consistent when we just look at user CPU time or just system CPU time instead.