Marvin McClain Jr.
Anderson Huang
CS 111

Benchmark Comparison of Simpsh, Bash, and Execline

A shell is a fundamental program that allows a user to interact with the underlying operating system of a computer. Many shells are readily available; as such, choosing between one shell or another ultimately comes down to issues of performance, features, and ease of use. In the following discussion, the performance capabilities of three shells - bash, execline, and simpsh - are tested. The specific benchmarks are discussed further below.

Benchmark No.	User Time(seconds)			System Time(seconds)		
	Simpsh	Bash	Execline	Simpsh	Bash	Execline
1A	0.5146	0.4922	0.4900	0.1726	0.1834	0.1820
	6.3346	6.442	6.4220	0.1762	0.2078	0.1940
1B	0.5931	0.6156	0.0000	0.3356	0.3572	0.1680
	6.7076	6.4234	0.0000	0.3957	0.4242	0.1780
1C	0.3184	0.3276	0.3360	0.0202	0.0272	0.0200
	0.4602	0.4778	0.4860	0.0856	0.1184	0.1140
2A	1.0748	1.1402	1.0780	0.0897	0.119	0.1140
	0.4202	0.4396	0.4220	0.0854	0.1154	0.1240
2B	0.7051	0.7182	2.1940	0.1196	0.1636	0.2360
2C	6.8852	7.6808	7.0540	0.2822	0.3334	0.3740
3A	14.0963	13.768	14.0360	0.0974	0.1032	0.1340
3B	6.5109	6.2708	6.3800	0.1093	0.1134	0.1140
3C	6.3884	6.297	6.3380	0.2806	0.2820	0.2740

Figure 1: A comparison of simpsh, bash, and execline using three benchmarks. They are measured against two metrics, user time and system time. Each shell ran a benchmark consisting of numerous tests. All shells executed the benchmark five times, and the times shown are averages of all trials.

Benchmarks:

To evaluate performance and specifically to see how these shells dealt with large files, we pitted these shells against 100MB files. These were created using the Linux devices '/dev/urandom' and '/dev/zero'. Benchmark 1 involves sorting, transliterating, and filtering through these large files (which is done using the 'uniq' command). Benchmark 2 continues testing of these shells' abilities to parse through large files with the 'sed', 'comm', and 'diff' utilities. And Benchmark 3 analyzes the shells' performance during the compression of files with 'bzip2', 'gzip' and 'tar'. Below is a summary of the tests and their descriptions.

Test No.	Description
1A	Sort 100 MB file, either with all zeros or random text
1B	Sort 100 MB file, sending data through pipes, transliterating text, cat command
1C	Filter 100 MB file with uniq command
2A	Use stream editor on 100 MB file
2B	Compare very large files
2C	Create a diff file given two large files
3A	Compress large file with bzip2
3B	Compress large file with gzip
3C	Compress large file with tar

Figure 2: Description of the tests used on each shell.

Analysis:

Aside from a few anomalies, the performance of each shell are relatively the same across the board. The pipe code used in measuring the performance of execline yielded unexpected results; each run of the benchmark gave a user time of 0.00s, which differs from the data obtained from the testing of the other shells. The final system time given by execline also deviates from that of bash and simpsh. Given our limited experience with execline, these differences in user and system times are most likely a result of syntax errors.

Conclusion:

Despite the discrepancies in the execline benchmark, by extrapolating from the rest of the data available, we predict that, provided correct syntax, all shells perform more or less the same on large scales. Time differences are on the order of tenths or hundredths of seconds, which are negligible. Further testing may need to be done to determine if more resource-intensive commands on our test files will cause the performance times of the three shells to exhibit more variance.