We did all our tests five times on server 7 on SEASNET, and we report the average of these five tests in our tables below. All the times are in seconds. The total times are rounded to 3 decimal places because they were timed with the `time` command, and simpsh's subcommands are rounded to 6 decimals because they used `getrusage`.

Test 1	user	sys
bash	1.024	1.589
simpsh	0.978	1.564
execline	1.028	1.615

Test 1	base64 /dev/urandom	head -c 20000000	sort	tr A-Z a-z	program total
simpsh: user	0.025209	0.005405	0.920761	0.023409	0.978
simpsh: sys	1.451232	0.027260	0.055754	0.029446	1.564

Test 1 tested each shell's performance when executing a command with pipes. Both user and system time were close for all tests, but simpsh was the fastest, and execline was the slowest. The resource usage for simpsh's subcommands, obtained from the "--profile" option, are shown in the second table above. The other shells' subcommand times were not measured because bash and simpsh cannot time piped subcommands.

Test 2	child 1	child 2	child 3	child 4	program total
bash: user	0.2042	0.2368	0.2444	0.2276	0.913
bash: sys	0.0158	0.0318	0.0306	0.0366	0.115
simpsh: user	0.209864	0.222565	0.219161	0.218853	0.871
simpsh: sys	0.021580	0.028767	0.032741	0.032922	0.118
execline: user	0.21	0.21	0.21	0.21	0.850
exeline: sys	0.00	0.00	0.00	0.00	0.039

Test 2 was a series of four child processes, each running the sort command on a 10 MB file in the background; each shell waited for these processes to finish. The average time of each child was recorded, plus the program's total time. In bash, we used the time command to calculate the resource usage of the four children, and we weren't able to do this for the program total so we added the four times to get the total. In simpsh, we used --profile to get the resource usage of the children and used the time command to get the program total. In execline, we used the time command to get resource usage of the children and the overall program. Looking at the program totals, execline had the lowest system time since it doesn't count the child processes' system time. Bash was the slowest shell, and execline was slightly more efficient than simpsh.

Test 3	user	sys
bash	3.648	0.135
simpsh	4.246	4.743
execline	3.782	0.210

For this test, we executed a somewhat fast program 50 times. Bash was the fastest, followed closely by execline. Simpsh was the slowest for this test. The overhead in system time for simpsh may have been due to needing to remember each subcommand's argv so that we could output it if "--wait" was given, as well as changing the standard file streams to /dev/null. Thus, malloc and free were called for each process, and dup2 was called three times for each process. Bash and execline have no need to do these operations required by simpsh's spec, so they have relatively little system time. Execline may have been slower for this test because the script was so long; execline passes the rest of the script into the first program's argv, which passes it into the second, and so on, so each process had to be given a large array for argv. The times of individual subcommands are not shown because they are all small, and there were many child processes.