A) 1 run per command

System : ls -- 1487 ms

System : ls -al -- 4344 ms

System : pwd -- 621 ms

System : echo $USER -- 3031 ms

System : cwd -- 591 ms

System : date -- 1107 ms

System : ps -- 15247 ms

System : exit -- 0 ms

Fork : ls -- 1851 ms

Fork : ls -al -- 6374 ms

Fork : pwd -- 618 ms

Fork : echo $USER -- 601 ms

Fork : cwd -- 202 ms

Fork : date -- 636 ms

Fork : ps -- 11739 ms

Fork : exit -- 0 ms

VFork : ls -- 1714 ms

VFork : ls -al -- 6494 ms

VFork : pwd -- 533 ms

VFork : echo $USER -- 567 ms

VFork : cwd -- 145 ms

VFork : date -- 588 ms

VFork : ps -- 16617 ms

VFork : exit -- 0 ms

Clone : ls -- 1841 ms

Clone : ls -al -- 11167 ms

Clone : pwd -- 682 ms

Clone : echo $USER -- 664 ms

Clone : cwd -- 7094 ms

Clone : date -- 2092 ms

Clone : ps -- 9122 ms

Clone : exit -- 13 ms

commands and time ranking:

1 - fastest

2 - runner up

3 - third place

4 - slowest

ls:

1 -System 2-VFork 3-Clone 4-Fork

ls -al:

1- System 2-Fork 3-VFork 4-Clone

pwd:

1-VFork 2-Fork 3-System 4-Clone

echo $USER

1 - Vfork 2-System 3- Fork 4-Clone

cwd

1 -VFork 2-Fork 3-System 4-Clone

date

1 -Vfork 2-Fork 3-System 4-Clone

ps

1 - Clone 2-Fork 3- System 4-VFork

exit

1 - System, Fork, Vfork 2-Clone

B) Average of 10 Runs Per Command

System: Fork: VFork: Clone:

Ls = 6171.8 (4) 4839.9 (1) 5019.5(2) 5401.5(3)

ls -al = 20495.0(4) 7661.7 (1) 8017.9 (2) 10688.7 (3)

pwd = 3441.9 (3) 2389.1(2) 2339.2 (1) 3018.1(4)

echo $USER = 3858.4 (4) 2531.9(2) 2774.2 (1) 2939.6 (3)

cwd = 7953.7(4) 1061.8 (3) 596.3 (1) 979.7(2)

date = 9359.1 (4) 3016.8 (2) 2862.0 (1) 3223.0(3)

ps = 42501.6(4) 31716.2(2) 30795.0 (1) 32201.4(3)

exit = 3.2(3) 2.6(1) 4.1(2) 14.9 (4)

Note: (#) - the ranking of the runtime of the command between the 4 versions

1 = fastest, 4 = slowest

C) Overall Average of Each version of tiny shell from the commands run

System: 11048.0 ms (4)

Fork: 6625.0 ms (2)

VFork: 6551.025 ms (1)

Clone: 7308.3625 ms (3)

Note: (#) - the ranking of the runtime of the command between the 4 versions

1 = fastest, 4 = slowest

D) Discussion and Notes:

VFork can be understood to have the overall fastest runtime because it does not need to create a copy of memory for the child process as Fork does nor does it need to call a child function and deal with overhead of passing a function.

Fork can be understood to be the second fastest because it does not need to deal with overhead of calling a child function as Clone does.

System is slower than clone because it contains more function calls compared to Fork, Vfork and Clone. Thus the results from these simple commands corresponds to the workings of each shell version and theory discussed in lectures relating to memory, overhead from different actions, and communication.

Since running 1 command as I did in part (A) and making comparisons in that way is not reliable because many different nuances can alter the runtime of processing a command.

Note, the timing from part (B) is different than part (A) because I am running my program from ssh on my laptop and I have different applications open. Part (A) was done on a Trottier computer.

Note on directing files or c programs t my tiny shell:

directing a file with commands on separate lines leads to continuous print of tshell>

directing a file will stop continuous prints of tshell> if it has exit

directing c program leads to continuous prints of tshell>

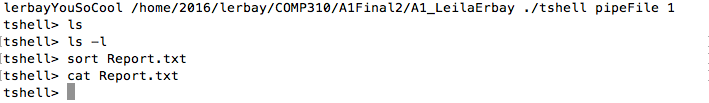
These have been noted and if time permitted I would have liked to determine how to fix these errors.

E) PIPE REPORT -- PROOF of WORKING FIFO (PICTURES)

My pipe works such that the user enters 1 and either the pipe file is created using the input name or uses the one already existed by checking the file name.

both processes must enter exit (they don't communicate through exit) to exit both tiny shells

Writing to FIFO:



READING from FIFO:

