Mistakes:

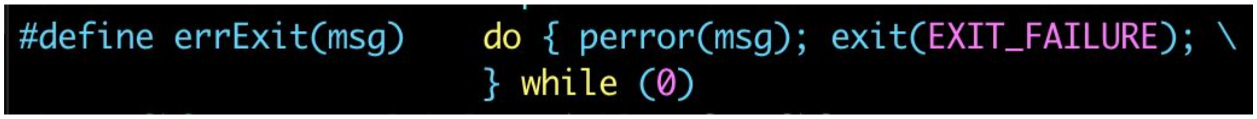
- No wait() called for the parent in the vfork() implementation. The parent only freezes until exec() or exit is called in the child. After that the parent continues. So for commands that expect continuous user input, the parent must wait().
- Return value of exec() only tells if the process loading was successful. Does not indicate if the loaded process crashed. So must use WIFEXITED() on the status to check this.
- Return value of exec() must be always checked. And if it failed, must call exit() or _exit() for the child to terminate. If not the child continues and acts just like the parent and waits to accept new shell commands.
- Many students did not understand what the PIPE based implementation was supposed to do. They tried to explicitly read from the PIPE and pass it to the other process. What was expected was to make the STDIN and STDOUT of each process to be the ends of the PIPE.
- Many students, re-wired the STDIN and STDOUT to point to the PIPE before calling fork(). In that case they have to clean up the wiring after the child process exits. If not whilst the parent on the write end holds the PIPE open, the child on the other end will not start reading and processing from the PIPE.
- A better approach is to call fork() and inside the child do all re-wiring.
- A signal must be always included as the last 8 bits of the flags to clone(). If not no signal will be sent to the parent when the child terminates.
- The idea behind getting cd to work - “cd” unlike other commands like “ls, pwd etc” is not a unix program. It’s a shell specific command. Thus, different shells have their own ways to implement it. So, to get it working you must run the command in a sub-shell. Or use chdir() to get it working.
- When implementing using chdir(), you must account for
  - Moving to HOME by just typing cd
  - Moving to HOME by typing cd ~
  - Moving within current dir and going backwards using .. and .
- Stack size must be reasonable. 512 is too small a stack size.
- Report did not have timing results for system() based implementation
- TIme measurements must be done over many iterations and averaged. Also across many different commands for comparison.

- Use memory allocation functions carefully, always specify the amount of memory you want in multiple of the data type for which you are allocating.
    - For example do not do this.
        - int *p = malloc(100)
    - Instead do this
        - int *p = (int *) malloc(sizeof(int) * n)

          Where n denotes how many integer elements you want to allocate.
- A good practice is to initialize a stack of type void for the clone () function.
-

<u>Good things:</u>

- Use of macro to print error message.



```
#define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \
                        } while (0)
```

- Used "perf" tool to report performance measurements.

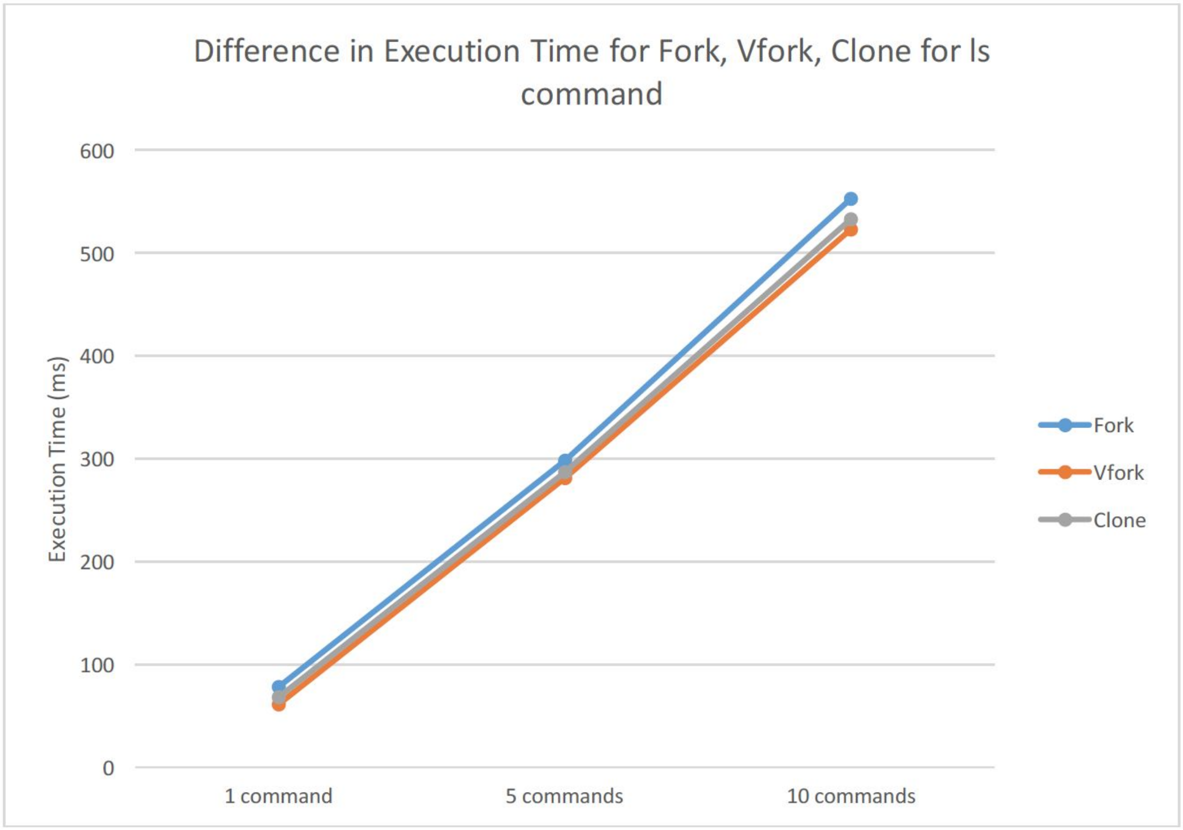
```
perf stat output for vfork

6.707514    task-clock (msec)      #   1.530 CPUs utilized
51     context-switches       #   0.008 M/sec
39     cpu-migrations         #   0.006 M/sec
1,064    page-faults            #   0.159 M/sec
22,244,807    cycles                 #   3.316 GHz
<not supported>   stalled-cycles-frontend
<not supported>   stalled-cycles-backend
14,873,275    instructions           #   0.67  insns per cycle
3,029,155    branches               #  451.606 M/sec
155,461    branch-misses          #   5.13% of all branches
0.004384381 seconds time elapsed

perf stat output for clone:

6.915397    task-clock (msec)      #   0.560 CPUs utilized
41     context-switches       #   0.006 M/sec
28     cpu-migrations         #   0.004 M/sec
1,231    page-faults            #   0.178 M/sec
22,992,040    cycles                 #   3.325 GHz
<not supported>   stalled-cycles-frontend
<not supported>   stalled-cycles-backend
16,301,965    instructions           #   0.71  insns per cycle
3,314,031    branches               #  479.225 M/sec
170,712    branch-misses          #   5.15% of all branches
0.012358182 seconds time elapsed
```

- Timing provided as graphs.



- Had printed a complete prompt like a real shell with hostname and username.
- Implemented "cd" to work with all sys_calls using chdir().
- Checking for "&" to see children running in the background.
- Perfectly handled signals
    - The parent ignore whilst the child is running and accepts when child is not running.

Command		System	Fork	Vfork	Clone (Vfork)	Clone (w/o Vfork)
ls	**Avg (µs)**	**1147**	**1183**	**1097**	**1168**	**1146**
	SD (µs)	**232**	**336**	**231**	**293**	**205**
with 50 fd / malloc of 10 000 int	**Avg (µs)**	**1170**	**1219**	**1171**	**1208**	**1217**
	SD (µs)	**253**	**435**	**394**	**266**	**342**
date	**Avg (µs)**	**835**	**832**	**822**	**875**	**852**
	SD (µs)	**387**	**197**	**282**	**217**	**286**
with 50 fd / malloc of 10 000 int	**Avg (µs)**	**833**	**870**	**839**	**913**	**959**
	SD (µs)	**152**	**235**	**238**	**475**	**286**
./crash-test	**Avg (µs)**	**95623**	**96455**	**95347**	**93684**	**94327**
	SD (µs)	**8964**	**5276**	**3264**	**5790**	**7311**
pwd	**Avg (µs)**	**352**	**353**	**308**	**361**	**358**
	SD (µs)	**114**	**119**	**97**	**108**	**119**
with 50 fd / malloc of 10 000 int	**Avg (µs)**	**357**	**357**	**325**	**392**	**381**
	SD (µs)	**103**	**108**	**140**	**219**	**156**
echo "Hello"	**Avg (µs)**	**397**	**400**	**342**	**398**	**403**
	SD (µs)	**224**	**202**	**143**	**183**	**200**
ps	**Avg (µs)**	**7379**	**7390**	**7384**	**6464**	**6278**

Averages:

system()	-	4.08 / 5.00	
fork()	-	8.39 / 10. 00	
vfork()	**-**	**3.80 / 5.00**	**(no wait or not capturing seg-faults)**
clone()	**-**	**15.69 / 20.00**	**(many failed to implement cd)**
pipe	**-**	**15.61 / 25.00**	**(incorrect rewiring, file-redirection fails)**
Timing	-	8.65 / 10.00	
Report	-	10.95 / 15.00	
Code Q	-	9.07 / 10.00	
Average	**-**	**74.84 / 100.00**	