1. Start with this program that takes in a command and executes it calling execv/execvp/execvpe, i.e. that creates a basic shell program. Change the program so it takes two commands at a time, then executes them both, serially as background jobs. Next, change it so it executes both commands concurrently in the background. Search google to discover what you might use to execute these jobs in parallel, if you know you have two cores/processors?

**Answer:**

The code to create a shell that takes in two commands at a time, then executes them both serially as background jobs can be found in the Task\_6 > Task\_6.1 folder (please feel free to check it out for more detail on my code). Now, after running that code on my virtual machine (which runs Ubuntu 18.24), the two commands were indeed being executed serially, one by one, as background jobs and the screenshot below proves the correct (expected) behavior of my code:

A screenshot of a computer

Description automatically generated with medium confidence

The code to create a shell that takes in two commands at a time, then executes them both concurrently in the background can be found in the Task\_6 > Task\_6.2 folder (please feel free to check it out for more detail on my code). Now, after running that code on my virtual machine (which runs Ubuntu 18.24), the two commands were indeed being executed concurrently in the background (as child 2 did not have to wait for child 1 to finish execution before it started to execute, or vice versa, but instead, both children were started/started working at the same time) and the screenshot below proves the correct (expected) behavior of my code:

A screenshot of a computer

Description automatically generated with medium confidence

Now, regarding what might be used to execute these jobs in parallel (perhaps on different cores/processors), there appears to be a lot of different options that we can choose from, each with its pros and cons obviously.

* The first tool we can use to execute shell commands in parallel is the GNU Parallel tool. Simply put, GNU Parallel can split a given input and pipe it into commands in parallel.[[1]](#footnote-1)
* We can also use Bash’s basic parallelization tools (& and wait) however those are not up to bar when it comes to more complex problem.
* Using Bash subshells which allow us to run parts of a script in parallel.[[2]](#footnote-2) In this case, the subshells will let the script do parallel processing, thus executing tasks in parallel.

1. https://stackoverflow.com/questions/5547787/running-shell-script-in-parallel [↑](#footnote-ref-1)
2. https://tldp.org/LDP/abs/html/subshells.html [↑](#footnote-ref-2)