Example: shells in shells

In your Docker setup, you'll be executing a short series of commands in order to better understand the correct behavior. We'll primarily be making use of two commands, `ps` and `jobs`. Recall that `ps` gives you information about all processes running on the system, while `jobs` gives you a list of jobs that the current shell is managing. Enter the following commands in your terminal, and you should see similar behavior:

```
workspace $ ps
PID TTY TIME CMD
20970 ttys002 0:01.30 bash
workspace $ sh
$ ps
PID TTY TIME CMD
20970 ttys002 0:00.63 bash
22323 ttys004 0:00.01 sh
```

At this point, we have started a `sh` shell within our `bash` shell.

```
$ cat
hello
hello
world
world
^Z
[1]+ Stopped(SIGTSTP) cat
$ ps
PID TTY TIME CMD
20970 ttys004 0:00.63 bash
22323 ttys004 0:00.02 sh
22328 ttys004 0:00.01 cat
```

Notice how sending a `CTRL-Z` while the `cat` program was running did not suspend the `sh` nor the `bash` programs.

After examining the output of `jobs`, stop the `cat` program with `CTRL-C`.

```
$ jobs
[1]+ Stopped(SIGTSTP) cat
$ fg
cat
^C
$ exit

workspace $ ps
PID TTY TIME CMD
20970 ttys004 0:00.65 bash
```

Since `exit` terminates the shell, we terminated the `sh` program. Enter `exit` again and your terminal will close.

Before we explain how you can achieve this effect, let's discuss some more operating system concepts.