1. Compile and run 3000shell.c
   * Done with gcc -O -g -Wall 3000shell.c -o 3000shell
2. Try running programs in the background using & after commands entered in 3000shell. What happens to the input and output of the program? Try this for simple programs like ls and bc. Then, try it for more complex interactive programs such as nano and top.
   * For simple programs like ls and bc it prints out the process id for running the program in the background and prints out exit status as well. And lets you do more than one command at a time
   * For complex programs such as nano it allowed me to edit in another file called & once I exit out of it, I am able to edit the original file I was calling to
   * Top ran in background wont let you scroll, for complex programs running them in the background messes it up
3. You may have trouble interacting with the shell after running programs in the background. How can you recover from such a situation?
   * You can use ctrl + d the process and terminate process and re run commands
4. Run 3000shell under gdb and observe all the system calls it makes using catch syscall (after setting a breakpoint on main so you don't see the syscalls when it starts).

Where does each system call happen? In what context (source and assembly)?

Consider both parent and child processes (by setting follow-fork-mode to parent and child).

Compare with the output of strace (e.g., run strace -fqo 3000shell.log ./3000shell). For info on gdb, see the [GDB quick start](https://homeostasis.scs.carleton.ca/wiki/index.php/GDB_quick_start).

* + Do break point at a function then run catch syscall in same terminal

1. 3000shell implements a simple form of output redirection. What syntax should you use to redirect standard output to a file?
   * ./3000shell > outputfile.txt
2. Why are lines 207-210 there (the check for pid == -1)?
   * Checking waiting status if the child pid dies it returns -1 as an error instead of 0,1, or 2
3. Make find\_binary show every attempt to find a binary.
   * printf(“%s\n,p); to show each attempt to find a binary
4. Make the shell output "Ouch!" when you send it a SIGUSR1 signal.
   * Graphical user interface, text

     Description automatically generated
     1. This was added to the signal handler function for instructions how to handle SIGUSR1
   * Text

     Description automatically generated
     1. This was added to register the SIGUSR1 handler
5. Delete line 324 (SA\_RESTART). How does the behavior of 3000shell change?
   * After sending a signal, instead of being able to run more commands, it exits the shell
6. Replace the use of find\_env() with getenv(). How do their interfaces differ?
   * Student is now null and not able to make any commands, they all don’t do anything
7. Make plist output the parent process id for each process, e.g. "5123 ls (5122)". Pay attention to the stat and status files in the per-process directories in /proc.
8. Implement redirection of standard error
9. Implement redirection of standard out for plist() (the same as if it was an external command).
10. Implement a built-in 3000kill command that works like the standard kill command. (What system call/library call is used to send signals?)