Note: My shared folder keeps having issues, and the manual could not fix it so I could not include screenshots of the files. But due to the simplicity of the code and the detail of the explanations, I do not imagine you will need the pictures. Sorry for the inconvenience.

1. After running both versions of the myprintenv.c program, one with the child process environment variables, and one with the parent process environment variables, I see that there were no noticeable differences. I created 2 separate executables, one for the child and one for the parent, for simplicity when comparing. In doing so, I had 2 shell output files as well, separated the same way. In looking through the files with my plain eye at first, I saw no differences. Then when using the "diff -q" command, it concluded that the two files did in fact differ. So, I then used the "diff -u" command to see the actual differences, and I saw only 3 differences:
   - The first difference was at line 26, where the identifier for the terminal screen was different between the two files:
     childOUTPUT:
     GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/e9413d3e_e726_4859_98f9_ff10f697d0db
     parentOUTPUT:
     GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/239065fa_cbc8_4ef3_9ad3_f903ec6b104b
     And this difference is just due to the difference between processes given their unique ID's.
   - The second difference was at line 34, where the service identifiers did not match, so this only tells us that these two are separate processes, nothing important.
     childOUTPUT: GNOME_TERMINAL_SERVICE=:1.419
     parentOUTPUT: GNOME_TERMINAL_SERVICE=:1.421
   - The third and final difference was at line 45, showing us that these two files are from 2 separate executables, as we already have assumed anyways.
     childOUTPUT: _=./childEXE.out
     parentOUTPUT: _=./parentEXE.out

2. In compiling the two versions of the file, it is concluded that execve( ) does not automatically inherit environment variables from the current process unless it is explicitly told to in the envp argument.
   - In the first instance of the file where "execve("/usr/bin/env", argv, NULL)", the program from /env is created in a clean, empty environment since envp=NULL, giving the process no inherited environment variables. So the output file was completely empty.
   - In the second instance where "execve("/usr/bin/env", argv, environ)", since envp=environ, the program is executed with the current process's environment variables. So, the output file was filled with all of the inherited environment variables from the current process.