Homework 2

Due Date: November 2, 2000

200 Points

1. (20 points) Chapter 14, exercise 6
2. (20 points) Chapter 14, exercise 9
3. (160 points) This exercise asks you to determine how the various shells access environment variables, and test for a potential problem.
   a. Write a program called `envalter` to add environment variables to an environment and then spawn a subprogram. Your program should take the following arguments:
      - `–b env` add the environment variable `env` to the beginning of the environment. `Env` may be an environment variable name or a name and value (`var` or `var=value`, respectively).
      - `–d env` delete all occurrences of the environment variable `env` from the environment. If `env` is an environment variable name, delete all environment variables with that name. If `env` is a name and value, delete only those variables with the given name and value.
      - `–e env` like `–b`, except the environment variable `env` is added to the end of the environment.

   Hint: use `execve(2)` to execute the program. Do not use `system(3)!

   b. Write a second program called `shell` that determines whether a given shell uses the first or last search path in the environment. This program should take the following arguments:
      - `–f dir` put the first `xyzzy` program in this directory (if not given, use the directory `xyzzy1`)
      - `–l dir` put the second `xyzzy` program in this directory (if not given, use the directory `xyzzy2`)
      - `shell` use the named `shell`

   Hint: Create two programs called `xyzzy`. One should print “it’s the first” and the other “it’s the last”. Use the program you wrote in part a to delete the current search path, and add two new search paths. The first adds a search path containing the directory with the first `xyzzy` to the front of the environment, and the second adds a search path containing the directory with the second `xyzzy` to the end of the environment. Then spawn a shell and see which program is executed.

   c. Bundle your programs into a distribution mechanism that works as follows. After un-taring the program, the recipient types “make” to compile (set up) both programs. The recipient can then type “`.check shell`” where `shell` is the name of a shell (either relative or full path name) and the program will print: one of:
   - `shell`: uses the first occurrence of the environment variable
   - `shell`: uses the last occurrence of the environment variable