Homework 1

Due Date: January 14, 2000 **100 Points**

1. (*15 points*) Chapter 1, exercise 8
2. (*15 points*) Chapter 1, exercise 9
3. (*15 points*) *Robust Programming* handout, exercise 8
4. (*15 points*) *Robust Programming* handout, exercise 12
5. (*40 points*) This exercise asks you to look at a standard UNIX C library for problems with robustness. Please write two programs that use functions from the strings library (the functions in that library are listed in the *string*(3) manual page). You are to call the functions in such a way that they cause the library function to crash, or generate unpredictable results. Turn in the following:

   a. To demonstrate "crashing," use *gdb* output to show that the crash occured *within* the strings library function. That is, the program must call to the standard I/O library function, but cannot return from that call.

   b. To demonstrate "unpredictable results," run your program (without changes) on at least two different types of computers in the CSIF (for example, once on a Linux system and once on an HP) and show that the results of the function differ (you can use *gdb*, or print the relevant values).

   c. In *both* cases, please explain why the library routine answers. Support your hypothesis with references to the *gdb* output (or to source code, if that is available—but state how you know your version of the source code is the version actually installed!)

   Please submit both the programs and typescripts for each program showing the crash or the unpredictable results. ***Important note: you must supply the correct type of argument for the functions. You may not, for example, pass a character pointer where a file pointer is expected.***

   Your two programs must not use the same technique to cause the problem. For example, if you discover a certain argument to *strcat*(3) causes a core dump, you cannot use that same argument to cause a core dump in *strcpy*(3). However, if a different argument will cause *strcpy*(3) to crash, feel free to use that.

   You must submit a Makefile that we can invoke to compile your programs automatically. The compiled programs *must* be called "str1" and "str2". When you submit your program, please place your programs, Makefile, README, and any other relevant files into a subdirectory. Please call the directory *string*. Generate a tar file called *string.tar*. Use *handin* to submit *string.tar*. If we need to testit, we will un-tar it, run *make*, and execute *str1* first, then *str2*.