Lab #5
Libraries
Due In Lab, October 1, 2003

Name: ____

Lab Time: ____

Grade: __________/10

Building and Installing Libraries

In this part of the lab you will build and install two simple libraries.

1. Copy the file `~csci3308/src/lab05.tar` into your tmp directory. Extract the contents of the tar file into your source directory and then delete the tar file.
2. Enter the newly created lab05 directory and examine its makefile.
3. What two library files will be created by this makefile?
4. Figure out what object files will be placed in each library, and look at the source files for those object files to find the *functions* available in each library.
5. What functions are available in the first library?
6. What functions are available in the second library?
7. Build the two libraries in a subdirectory of the *build* directory and install the libraries and header file in the appropriate directories. You can either copy the makefile to the lab05 build directory or invoke make using the -f option to specify the makefile in the lab05 src directory.

8. What is the path to the installed libraries? (Be sure to specify the path abstractly using environment variables in the appropriate places. This applies to the questions below as well.)

9. What is the path to the installed header file?

Experimenting with Library Searches

In this part of the lab you will be experimenting with how the compiler selects object files from a library. You will build different versions of a simple application called `facts` that uses the funny facts library. The compile command will make use of the `-I`, `-L`, and `-l` options. The compiler does not recognize `~` as your home directory, so for the `-I` and `-L` options use `$HOME` instead.

- The `-I` option tells the compiler where to look for include files.
- The `-L` option tells the compiler where to look for library files.
- The `-l` option tells the compiler what libraries to use. The library name is given in a shorthand format. The compiler takes the name after the `-l` option, adds `lib` to the beginning, and `.a` to the end, and looks for a library of the resulting name. For example `-lm` means look for a file named `libm.a`. `libm.a` contains the standard Unix math libraries. Note, If you invoked the compiler with the flag `-llibm.a`, then the compiler would look for the file `liblibm.a.a`, which would most likely fail. Because of this convention, the `-l` option requires that library files begin with `lib` and end with `.a`.

Now you will compile the program `facts` from the libraries, and the source file `facts.c` which was included in `lab05.tar`. Rather than use a makefile to build the various versions of `facts` you will type the compile command directly. Also, you will compile from the source file `facts.c` directly to the executable file without creating a `.o` file. The general form of the command line is

```
gcc facts.c -o facts -IIncludeDir -LLibDir -lLibrary1 -lLibrary2
```

10. `facts` is a simple program meant only to test out the process of linking libraries. As such you will probably not use it after today, so you may not want to install it in your bin directory. For this lab feel free to compile

and run `facts` in a `tmp` directory. This is a case where you may want to use an architecture specific `tmp` directory, especially if you want to test out `facts` on several architectures at the same time.

11. First, build `facts` and link in only the library `libFunnyFacts1.a`.
12. What did you specify for the `-I`, `-L`, and `-l` options?
13. What is the output when you run the program?
14. Build `facts` again, but this time link with `libFunnyFacts2.a`. Why was there an error message?
15. Try again, but link with both `libFunnyFacts1.a` and `libFunnyFacts2.a`. Why did you not receive the same error as when you compiled with just `libFunnyFacts2.a`?
16. What is the output when you run the program?
17. Compile `facts` again, but reverse the order of the two `-l` options. What is the output when you run the program?

18. Why is it different from the previous version?

19. There is no way, without changing the library files, to compile `facts` so that it produces the output "`The national bird is the bald eagle`." Why is this?

20. Assume you are going to modify the makefile to create a library that will allow the facts program to say "`The national bird is the bald eagle`." Assume this new library would take its object files from a macro called `OBJFILES3`. Write the definition of this macro below.