CS 404: Problem Set 2 Answer Key

Directions for Submission

E-mail your answers to me at `ajp9@cornell.edu`. The subject of your message should be "CS404 PS2," and the body of the message should contain your answers. Some mailers can format messages using HTML or RTF. Please turn this feature off and send your message using plain text. If for some reason you cannot send your message as text, you may attach your answers as a text file.

Essential Knowledge–Please give a brief answer (1-2 sentences) for each

1. Why is it easier to link a C program which calls FORTRAN routines using a FORTRAN compiler? If you wanted to do the linking with the C compiler, what would you need to do?

   FORTRAN compilers hide the fact that they link to a library containing the FORTRAN intrinsic functions. By linking with FORTRAN, we save ourselves the trouble for hunting around the system for the library with the FORTRAN intrinsics.

2. Computer memory is one dimensional. How would the 2D array:

1	2	3	4
10	20	30	40
100	200	300	400

   be stored in the memory of a C (or C++ or Java) program. How would Matlab or FORTRAN store the array?

   C: [1, 2, 3, 4, 10, 20, 30, 40, 100, 200, 300, 400]
   FORTRAN: [1, 10, 100, 2, 20, 200, 3, 30, 300, 4, 40, 400]

3. You're given a version of LAPACK that has been compiled and stored in a Windows dynamically-linked library (DLL). What two things must you do to call routines in this library?

   You must link your program to the DLL's .LIB file. In order for you program to run, the DLL must be in your PATH. You can either move

the DLL into a directory in your path or you can add the directory containing your DLL to PATH.

4. True or False. If you don't register you code with the Library of Congress (and pay the $35 fee), someone may legally copy your code without your permission. Answer "true" or "false" and give a short explanation.

   FALSE: All original works are automatically subject to copyright protection as soon as they are placed in a fixed-medium (e.g. saved to your hard drive). Registering your copyright merely provides a paper trail certifying the originality of your work. This guards against someone stealing your code and claiming you copied it from them.

Windows DLLs or UNIX shared-objects

On Friday, I described how to create and link to DLLS and shared-objects. Your task is to build either a DLL or a shared-object library containing the routines in "system.c" from the RAD1D example (use the code available on www.cs.cornell.edu/Courses/cs403/2003sp/Lecture09/note.html).

5. Build your DLL or shared-object library, attach it to your e-mail and send it to me (also include the .exp and .lib files if you choose the DLL option).

6. Assume that I have the RAD1D source code except for "io.c." Tell me how to build and run RAD1D using the library you sent me.

   DLL: Using Visual Studio, you need to create two projects, a "Win32 DLL project" which I'm calling "radsystem" and a "Win32 console application" which I'm calling "rad1d." Add system.h and system.c to radsystem and place "__declspec(dllexport)" in front of all of the subroutines in system.c and also in system.h. Then build. This should create a .dll, .lib, and .exp file.

   Next, add all of the source files to the rad1d project except for system.c (system.h is needed to adhere to strict ANSI C style, but the decclspec stuff should not be necessary). Configure the "Additional Library Path" to point towards the location of radsystem.lib and add radsystem.lib to the list of "Object/library modules." Then build. This should create rad1d.exe.

Finally, to run, you need to configure your path to point towards radsystem.dll. You can either move radsystem.dll into your path (may require administrator privileges) or set your path to point towards the dll.

SO: To build a shared object library on ACCEL, I created a directory inside my rad1d directory called "Lib" and moved system.c and system.h into it. Inside the Lib directory, I compiled system.c to "Position Independent" object code by typing "gcc -c -fPIC system.c." This creates system.o. I then built the library using the linker:

ld -shared -soname libradsystem.so.1 -o libradsystem.so.1.0 -lc system.o

This creates the libradsystem.so.1.0 library. I'm a little fuzzy on what the version numbers mean, but to get it to work I had to copy libradsystem.so.10 to libradsystem.so.

In the main directory, I deleted any reference to system.c and system.o from the Makefile. On the line that builds the rad1d, I added "-L Libs" to tell the linker to look in Libs for libraries and then added "-lradsystem" to link to the shared object.

Finally, to run, I had to add Libs to my LD_LIBRARY_PATH by adding the following line to my .bashrc file:

export LD_LIBRARY_PATH= $(LD_LIBRARY_PATH):

/home/ajp9/rad1d/Libs