

# CS485/ECE440: Homework 2

Added requirement for output in each program

September 11, 2008, and 9/18/2008

## General

- **Follow the instructions below very carefully!** Name the files exactly as spelled out in the text below, follow the rules on file formats, etc. With this many students, we need to automate the process of grading as much as possible. For every minute we spend on each homework, we lose an hour! Please be considerate.
- This homework is about material covered in Chapter 1 and 2 of the text book.
- The answers to this homework are **due September 23, 2008**.
- Submit via e-mail to `riesen@cs.unm.edu` (Mail it before class on the 23rd.) Your subject line must say: "Homework 2 Submission" (and nothing else).
- Your answers to the questions below must be in a PDF or plain text file attached as file named `Homework2.pdf` or `Homework2.txt` according to its format. (Not `Hwk2.pdf`, and no Microsoft Word or image files, such as jpeg, please.)
- Keep your answers (and programs) succinct.
- Obey the University rules on plagiarism. In particular, do use libraries and the web to find information you need to answer the questions, but do not copy whole answers or programs. Reference your sources. The work you turn in must be *your* work.
- You are allowed (encouraged) to use the example programs on the class web site as a starting point.
- Your programs must be written in C or C++.
- Name your files as specified in the exercise below.
- Your programs must compile without warnings under gcc with the flag `-Wall` set.
- Provide a Makefile that compiles your programs assuming gcc and the gnu libraries are installed on the system.
- Create a compressed tar archive with the Makefile and the source code only (no `.o` files, executables, or READMEs).
- Name the tar archive `Homework2.tar.gz` (For this homework you will submit two attachments: `Homework2.pdf` and `Homework2.tar.gz`)

- Make sure your full name appears in the body of your message. Some email aliases do not easily translate to your name in Loboweb.
- Do not put any requests or messages into the body of your email. It is quite possible that we will never look at the body of your message; just the attachments. If you have questions or comments, send them in a separate email with a subject line other than “Homework 2 Submission”
- Your program code must be clean and readable. Be consistent with you indentation, use descriptive names for your variables and constants, etc. Make sure the code you send is plain ASCII text and formatted the way you intended. This is especially true if you are using an IDE which often produce messy source files. Use a plain text editor to look at your files. Poorly written code, even if it works, will loose some points.

**Exercise 1:** Transmit time

Do exercise 1.22 on page 59 in the textbook.

**Exercise 2:** Packet versus circuit switched

Do exercise 1.25 on pages 59 – 60 in the textbook.

**Exercise 3:** Encoding

Do exercise 2.1 on page 151 in the textbook.

**Exercise 4:** 4B/5B

Do exercise 2.2 on page 151 in the textbook.

**Exercise 5:** Byte stuffing

Do exercise 2.8 on page 152 in the textbook.

**Exercise 6:** Programming exercise

Write three small programs named `client1.c`, `client2.c`, and `server.c` (or `client1.cc`, `client2.cc`, and `server.cc` if you are using in C++).

The first client must send a set of 10 different floating point values to the server. You may choose the values of these 10 numbers and hard-code them into the first client.

The server must receive ten floating point numbers, add them up, and store the result. When contacted by the second client, the server must send it the calculated result. If no data is available yet, the second client must block until the server has received the data from the first client.

Use 5000 plus the last three digits of your student ID as the port number. Increment by one four each additional port you may need.

The clients take a hostname as their only argument. That argument is the host name where the server resides (may be `localhost` or `127.0.0.1`). The server does not take any command line arguments.

You may use the example program from Chapter 1 as a starting point. It is provided on the class web page.

Client 1 should print, one per line, the numbers it is sending to the server. Client 2 should print the result received from the server, and the server should print the numbers it received plus the result it computed.