CS244a
Winter 2006
Professor McKeown

Handout #1

CS244A: An Introduction to Computer Networks

Course Logistics and Grading

Instructor: Professor Nick McKeown
Office: Gates 340
Email: nickm@stanford.edu
Office Hours: Check web page
Class Web Page: http://www.stanford.edu/class/cs244a
Class Newsgroup: su.class.cs244a

Notes on email and questions: There are two ways to ask questions to the TAs. We prefer that you post your question to the class newsgroup. Each day, one or more TAs will be on duty to answer posted questions. They are offering you a guarantee: they will respond to all questions on the newsgroup within 24 hours during the week, and within 48 hours at weekends. On the weekend prior to an assignment being handed in they will try to respond within 24 hours. By using the newsgroup, everyone gets the benefit of reading the replies, and it helps to reduce the number of repeat questions.

For questions that are not suitable to be posted to the newsgroup (e.g. questions about your grades), you can email your TA directly. There is no guarantee of when you will receive a reply. I have asked the TAs to keep the average response time to under 48 hours, but I have asked them to give priority to answering questions posted to the newsgroup. We really want you to use the newsgroup :-)

Please don't *ever* send email to the class mailing list, or the TA mailing list — there are so many students in the class that we'd be flooded with emails!

Teaching Assistants:
(a.) Ari Greenberg (arigreen@stanford.edu).
(b.) Matt Falkenhagen (mjf@stanford.edu).
TA Office Hours are posted on the class web page. All office hours will be held in Sweet Hall.

Exams: There will be one in-class midterm and a final exam. Please visit the web page for location details for the final exam.

Final Grade: Midterm (15%), Final (25%), Programming Assignments (45%. Breakdown by assignment is #1: 10%, #2: 10%, #3: 15%, #4: 10%), Problem Sets (15%).

Taking Notes: I'll be using a combination of handouts and the overhead camera. When I use the overhead camera, I recommend you take notes. When I use handouts, I will make them available to you. All handouts are posted on the class web page, and hard copies will be available in class.

Prerequisites: CS140 is a strict prerequisite for CS244A, and will be enforced without exception. If you have not taken CS140 at Stanford, you need my permission to take CS244a. Send an email to nickm@stanford.edu. Include a description and/or a URL to the class you took at your undergraduate university. If you do not have permission to take the class, you may be dropped automatically. You need a strong background in C programming and familiarity with the Unix operating system. Take a look at Homework #1 (listed on the Timetable page) to get an idea of the type and quantity of work the first homework will require.

Taking two classes in the same time-slot: Occasionally I get asked to provide a makeup exam in a different time-slot for students taking two or more classes at the same time. Please note that the class is too large to accommodate different conflicts, and so *I don't provide makeup exams*. If you are taking two classes at the same time, you need to check in advance that the other class provides makeup exams.

Required Textbooks:
(a.) Bruce S. Davie and Larry L. Peterson, "Computer Networks: A Systems Approach", **3rd Edition**, 2003.
(b.) W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff, "UNIX Network Programming, Volume I: The Sockets Networking API", **3rd edition, 2003**.

Programming Assignments: There will be three UNIX network programming assignments.
Assignment 1: "FTP Client" Assigned: Tue 6th January. Should take no more than 20hours.
Assignment 2. "Your own router".
Assignment 3. "Your own TCP". Toughest assignment!
Assignment 4. "Your own complete stack".

Here is a summary of the requirements for all submitted programming assignments:
(a.) All programs must be written in ANSI "C". To make the grading more uniform, we can't except assignments in C++, Java, perl, ...
(b.) All programs must compile and run correctly using *gcc* on the solaris machines (e.g. ssh elaine43.stanford.edu). You are strongly encouraged to use *gdb* to debug your programs.
(c.) We want you to write good clean code, with no bad memory references. To this end, we provide *purify* to help you debug and write good code. It's an excellent tool, that will save you lots of time. I strongly encourage you to use it throughout your code development. Details are provided in the first assignment, including the ASCII interface for students logging in externally. To encourage you to use it, we require that you run *purify* before submitting your assignment, and that it indicates no errors in your program.
(d.) Additional requirements will be specified in each assignment.
For more details about the programming guidelines, refer to the web page:

http://www.stanford.edu/class/cs244a/project_guidelines.html

You can also find some Coding Guidelines that will explain what we are looking for, including some good and bad examples:

http://www.stanford.edu/class/cs244a/CS244aCodingGuidelines.html

Grading of Assignments: Projects are graded on a 10 point scale, of which 7 points represent how close the solution comes to correctly solving the problem, 1 point is devoted to whether code is properly commented and variables have mnemonic names, and 2 design points indicate how effectively the code solves the problem (proper choice of algorithms, clean program structure, etc). We go to great lengths to ensure that the grading is fair and consistent:
(a.) First of all, each assignment has a carefully defined set of grading guidelines that are agreed

upon by all the TAs. The grading guidelines will be available on-line to help you understand how your assignment will be graded.
(b.) We use grading scripts to test some aspects of your solution. We let you run the grading script too. This helps you test your code, and understand our expectations. More details about this in Assignment #1.
(c.) The grading script can test some, but not all, of the grading criteria, and so your TA will run some manual tests as well.
(d.) Finally, your TA reads through your code to understand the cause of some of the bugs (if any), to determine how well your code is structured, to see whether you used sensible variable names, etc.
(e.) Throughout the grading process, the TAs meet regularly to discuss and refine the way they grade and to improve consistency.
(f.) You will each be assigned to a particular TA for the duration of the class. Your TA will grade your first assignment so you get to know each other. For each successive assignment, your solution will be graded by a different TA.
(g.) For each assignment, we statistically adjust for variations among the TAs. Having said that, since we instigated the process above, we have found that there is no measurable variation in the grading. But we always check, just to be sure.

Late Policy: We try hard to balance two constraints. On one hand, there are a lot of students in the class and we need to be strict about deadlines. If assignments are not turned in on time, it delays the grading of your work, and your TA has to juggle grading different assignments at different times. I've found that this makes the class unmanageable, and your TA is left with less time to answer questions and help you with your assignments. On the other hand, emergencies happen from time to time. So here is the policy:
If you hand in an assignment 0-24 hours late, you will lose 25% of the grade. 24-48 hours late and you will lose 50% of the grade. 48-72 hours late you will lose 75% of the grade. After 72 hours you will receive a zero.

Free Late: Despite the tough stuff above, you have **one** free late of 24 hours. You can use it whenever you wish, but you must email your TA before the deadline to use your free late. Just like above, the 24 hours is a hard deadline, with no further extensions. We hope this lets you manage your quarter.

Exceptional circumstances: Please do not ask your TA for extensions as they are not authorized to grant them. If you have a medical emergency, then email me to discuss whether I can grant an extension. At times there may be system problems with the Sweet Hall cluster which may prevent you from completing your assignment on time. In these cases an extension will be granted to the entire class**.**

SITN students: SITN/SCPD students are required to follow the same deadlines as the in-class students. I do not give extensions because of travel requirements, or because of deadlines you have at work. You must schedule your work commitments so that you can complete the assignments, problem sets and exams just as our campus students do. There is no 7-day grace period in this class. For this reason, you are strongly encouraged to watch the Stanford-Online videos. You are required to follow the same programming guidelines at the in-class students. You will be required to login to your Sweet Hall account from your location in order to test your programs prior to submitting them. Local SITN students should come to class for the midterm and final exams. When you need to turn in your problem set, you need to give it to the SITN courier on the same day as the in-class deadline.

CS244a and the Honor Code

In previous years, several students taking CS244a have been found guilty of violating the Stanford Honor Code. In this course, the Honor Code is taken seriously and it is expected that all students will do the same. The good news is that the vast majority of students do take the Honor Code seriously. The bad news is that historical evidence indicates that some students will submit work that is not their own, shortchanging not only their own learning, but undermining the atmosphere of trust and individual achievement that characterizes Stanford's academic community. To protect academic integrity and the interests of all students, the course staff will investigate all possible Honor Code violations and refer them to the Office of Judicial Affairs as necessary.

I'm pleased to report that for three years in a row, no-one was found to have violated the Honor Code in CS244a. Please help me make it four good years in a row! If you have any questions or doubts about the Honor Code, please come and talk to me. Honor code violations are no laughing matter at Stanford and it is much better to ask what might seem like a silly question now than to risk your academic career. The Honor Code has a long tradition at Stanford dating back to Spring 1921 when the University first adopted the honor system. Today the Honor Code continues to govern academic conduct of both students and faculty at Stanford. The Honor code reads as follows:

THE STANFORD UNIVERSITY HONOR CODE

(a.) The Honor Code is an undertaking of the students, individually and collectively:
  (i) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
  (ii) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

(b.) The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

(c.) While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

The underlying premise of the policy is that all academic work represents independent, original work of the author and the Honor Code aims to foster an academic environment that encourages adherence to these principles. As we are all bound to respect and uphold the Honor Code, it is important to define acceptable and unacceptable behaviors with regard to this course so as to eliminate any ambiguity.

Permitted Collaboration: The following items are encouraged and allowed at all times for all students in this class:

- Discussion of material covered during lecture, problem sessions, or in handouts
- Discussion of the requirements of an assignment
- Discussion of the use of tools or development environments
- Discussion of general approaches to solving problems
- Discussion of general techniques of coding or debugging
- Discussion between a student and a TA or instructor for the course

Collaboration Requiring Citation: Two students engaging in more detailed discussions must be careful to document their collaboration. Students are required to include the names of those who provide specific assistance to properly credit their contribution, in the same manner as one would cite a reference in a research paper. The expectation is that even with a citation, the author must be able to explain the solution.

Some examples of collaboration that require citation include:

- Discussing the "key" to a problem set or programming assignment. Problem set questions are often designed such that the critical concept takes careful thought and gaining that insight from someone else must therefore be documented.
- Discussing the design of a programming project. Design is a crucial aspect of the programming process and discussion can be valuable. Any design input received from others must be cited.
- Receiving assistance from another student in debugging code. While the TAs are the preferred source for advice, any detailed assistance from someone else must be credited.
- Sharing advice for testing. For example, if someone provides important information on lessons learned ("my program didn't handle the case where the value was 0") that source must be credited.
- Research from alternative sources. Researching related topics, such as through the
- Internet, must be documented if the solution submitted is derived from the research information.

Unpermitted Collaboration: All submissions must represent original, independent work. Some examples of activities that do not represent original work include:

- Copying solutions from others. In particular, do not ask anyone to provide a copy of his or her solution or, conversely, give a solution to another student who requests it. Similarly, do not discuss algorithmic strategies to such an extent that you and your collaborator submit exactly the same solution. Use of solutions posted to websites, such as at other universities, is prohibited. Be aware that we photocopy some of the exams prior to handing them back.
- Using work from past quarters. The use of another student's solution or the posted class solutions from a previous quarter constitutes a violation. We use a sophisticated tool that cross-checks every assignment against every other assignment submitted this year, and previous years. It catches common code, even if comments and variable names are changed. In fact, in order to "fool" it, you have to change so much code that it would be quicker to do the assignment yourself. Developing good problem set questions and programming assignments often takes years and new assignments invariably have problems and that require polishing. To provide the most effective exercises, questions and assignments are commonly reused. Students retaking the course are expected to notify the course staff to avoid coming under suspicion.
- Studying another student's solution. Do not read another solution submission whether in electronic or printed form, even to "check answers."
- Debugging code for someone else. When debugging code it is easy to inadvertently copy code or algorithmic solutions. It is acceptable to describe a problem and ask for advice on a way to track down the bug.

This section was based on a handout from Tom Fountain, who teaches EE182 at Stanford. Some portions are based on similar collaboration policies written by Eric Roberts, Julie Zelenski, and the Computer Science Department at Brown University.