CS 473/573 Computer Networks
Spring 2015 Syllabus and Information

Table of Contents: **table of contents**, **what we will do today**, **assignments**

Instructor Directories: cd ~jkinne/public_html/cs473-s2015/

Textbook/course we will follow: Computer Networks MIT Course

Jeff to do:

-

You to practice:

- play with bitmap.c
- try out windump or tcpdump on your computer.
- hw1, hw2 - if didn't get them, then study and understand, and recreate on your own time sometime. Want motivation? They could show up on a quiz/test.
- chatServer.c, chatClient.c - improve if you want. try to make a simple game out of them (tic-tac-toe, x's and o's, or some other simple game). Or, Caesar cipher the data.
    - Note: you can use the x's as clients. ssh x1.indstate.edu and so forth.
    - Note: your port number is in BB, when you're playing around running servers and clients, make your program use your port number ...
- scan_input_shorter.c - modify so it also prints out the longest word in the file.
    - hint: need to count length of current word, and also need to keep track of longest word seen so far.

Table of Contents

Table of Contents

Table of Contents

General Information

Contact Your Instructor

Name: Jeff Kinne
Email: jkinne@cs.indstate.edu
Phone: 812-237-2136
Office: Root Hall, room A-129

Lecture, Exam, Office Hours

Lecture: Tuesdays and Thursdays from 11:00-12:15pm in Root Hall, room A-017.

Exam: Thursday, May 7, 10-11:50am in A-017.

Instructor Office Hours: I am generally in my office and available most MWF's from about 8:30am-4pm, but not until 10:30am on Mondays. My official office hours are Wednesdays 9:30-11:30am.

GA Tutoring: We have a few graduate assistants who will be in the computer science unix lab, room A-015 in the basement of Root Hall, for about 20 hours per week in total. You can go to this lab to work on your programs. The computers are unix machines, and you can use the cs473xx login that will be sent to you during the first week of class to use them. Or, you can bring your laptop to work on. Either way, you can ask the graduate assistants to look at your programs, and you can work with any other CS 473/473 students that are there (you could use the lab as a regular meeting place to work with your classmates). The regular hours that the lab will be open will be posted to **here on the department's website**.

Website: this google doc, or find a link from kinnejeff.com

Prerequisites

CS 202 with a C or better.

Required text

None. We will use online sources. For much of the semester we will follow along with this Computer Networks MIT Course.

Table of Contents

Course Announcements

Announcements regarding the course will be made both during class and via email to your @sycamores.indstate.edu email address. You should regularly check this email account or have it forwarded to an account that you check regularly. You can set the account to forward by logging into your indstate.edu email from Internet Explorer (the "light" version of the webmail client that opens up from Firefox or Chrome does not give the option to forward email).

Classroom conduct

You may not use cell phones, iPods/music players, etc. during class. You should be civil and respectful to both the instructor and your classmates, and you should arrive to class a few minutes before the scheduled lecture so you are ready for lecture to begin on time. You may use your computer during class if you are using it to follow along with the examples that are being discussed. You may not check email, facebook, work on other courses, etc. during class.

Course Description

The official description of this course from the catalog is

“The course is an introduction to networking and includes detailed study of Internet protocols and socket programming. Topics include a study of IP, UDP, and TCP protocols, as well as application layer protocols such as HTTP and SMTP. Students learn to program both a client and a server.”

In other words, the goal is to understand how computer networks work and be able to use them. This can be studied from a variety of levels - hardware and its configuration, various levels of software/programming, and the use of applications. We will look at each level a bit, but will focus most on the level of writing C/C++ programs to communicate via the internet.

Course Outline

We begin the course by following along with this Computer Networks MIT Course. Along the way, we will get a view of each of the following topics.

1. Protocols, Sockets, TCP/IP.
2. Socket Programming.
3. TCP/IP Protocols in Detail.
4. Application Level Protocols.

Table of Contents

Grading and Assignments

The students of this course have the following responsibilities: read assigned readings before lecture, attend lecture, complete homework assignments, take in-class quizzes, take exams, and complete a project. The final grade consists of:

- **Project: 15%** of the final grade.
- **Homeworks and Quizzes: 30% total**. Most weeks there will be at least one homework assignment or quiz.
- **Exams: 45% total**. There will be 3 exams. The first is worth 10%, the second is worth 15%, and the final exam is worth 20% of the total final grade.
- **Class Attendance/Participation: 10% total**. Attendance will be taken at the beginning of each class. Half of your attendance/participation score will consist solely of whether you were present when attendance was taken each day - the total number of days present divided by the number of lectures in the semester. The other half of your attendance/participation grade will be assigned at the end of the semester based on how attentive you were in class throughout the semester.

Late Homeworks

All homework assignments will be given a preferred due date. Assignments can be turned in past the preferred due date, but any assignments turned in late will have their value multiplied by 80% (so the highest grade you can get on a late assignment is 80%). Each assignment will have a “final due date” past which no credit will be given.

Start Homeworks Early

I suggest attempting a homework assignment the day it is given, or the day after, so that if you have a problem you can ask early. If you continue to have problems in trying to complete the assignment, you will have time to ask again. Many of the homework assignments require thought and problem solving, which takes “time on the calendar” not just “time on the clock”. By that I mean that spending an hour on 3 consecutive days is likely to be more productive than trying to spend 3 hours at once on the assignment.

Expected Amount of Work

My expectation is that an average student will spend about 4 hours OUTSIDE of class each week (that is in addition to class time) WORKING PRODUCTIVELY/EFFICIENTLY (not just

staring at the computer) to complete their coursework for this class. Some students may spend less time than this, and some students will spend more.

Grade Cutoffs

I will design homework assignments and exams so that a standard cutoff for grades will be close to what you deserve. After the first exam I will create a grade in Blackboard called "Letter Grade" that is what your letter grade would be if the semester ended today. Initially, I will assign the following grades: 93-100 A, 90-93 A-, 87-90 B+, 83-87 B, 80-83 B-, 77-80 C+, 73-77 C, 70-73 C-, 67-70 D+, 63-67 D, 60-63 D-, 0-60 F

My goal is that the different grades have the following rough meaning.

A+/A
You understand everything and probably could teach the course yourself.

B+/A-
You understand nearly everything, and should be all set to use this knowledge in other courses or in a job.

C/C+/B-/B
Some things you understand very well and others you don't (more towards the former for a B and more towards the latter for a C).

D-/D+/C-
You did put some effort in, and understand many things at a high level, but you haven't mastered the details well enough to be able to use this knowledge in the future.

F
Normally, students that get an F simply stopped doing the required work at some point.

Blackboard

The course has a blackboard site. Click here to go to blackboard. You should see this course listed under your courses for the current term. The blackboard site is only used for giving you your grades. All course content, schedule, etc. is kept in this google doc (which you are currently viewing).

Academic Integrity

Table of Contents

Please follow these guidelines to avoid problems with academic misconduct in this course:

- **Homeworks:** You may discuss the homework assignments, but should solve and finish them on your own. To make sure you are not violating this, if you discuss with someone, you should DESTROY any work or evidence of the discussion, go your separate ways, SPEND at least an hour doing something completely unrelated to the assignment, and then you should be able to RECREATE the program/solution on your own, then turn that in. If you cannot recreate the solution on your own, then it is not your work, and you should not turn it in.

- **Note on sources:** if you use some other source, the web or whatever, you better cite it! Not doing so is plagiarism.

- **Exams:** This should be clear - no cheating during exams. The exams will be closed-book, closed-notes, no computer, and no calculator.

- **Projects:** You should not copy from the internet or anywhere else. The project should be your own work. It will be fairly obvious to me if you do copy code from the internet, and the consequences will be at the least a 0 on the project.

If cheating is observed, you will at the least receive a 0 for the assignment (and may receive an F for the course), and I will file a Notification of Academic Integrity Violation Report with Student Judicial Programs, as required by the university's policy on Academic Integrity. A student who is caught cheating twice (whether in a single course or different courses) is likely to be brought before the All-University Court hearing panel, which can impose sanctions up to and including suspension/expulsion. See the Student Code of Conduct and Academic Integrity Resources for more information.

Please ask the instructor if you have doubts about what is considered cheating in this course.

Special Needs

If you have special needs for the classroom environment, homeworks, or quizzes, please inform the instructor during the first week of classes. If you have any such needs, you should go to the Student Academic Services Center to coordinate this. See Student Academic Services Center - Disabled Student Services for more information.

Table of Contents

Assignments

Assignments will be posted to this document and announced in class. Things will be listed here most recent first.

Reading/Viewing Assignments

Reading/viewing assignments are listed according to their DUE DATE. That means you should have that reading or video viewing done before class on that day.

- 2/3: See links in “What we will do today”
- 1/29: L3, T1 from MIT networks course
- 1/22: L2 from MIT networks course
- 1/15: L1 from MIT networks course

Exams

First Exam

- max(paper, computer) is worth 65%. min(paper, computer) is worth 35%
- IPv4/TCP headers - I’ll give a printout of what those are.
- Paper component - just you, paper, pencil/pen, do paper part first
    - Answer questions about IPv4/TCP headers.
        - Given a header, what is the BLANK field?
        - What is the maximum/minimum possible BLANK field (out of the ones we’ve talked about)?
        - Given a header, which field doesn’t make any sense (out of the ones we’ve talked about)?
    - memorize the return types, parameters, and basic use of:
      printf, scanf, strlen, strcmp, strtok, fopen, close,
      functions listed under socket programming
        - For example, I give you a line that uses one of the functions, and you say what each one means.
        - Given some code fragment, add a line to accept a socket connection.
        - What is the 3rd parameter to BLANK function? What does BLANK function return?
    - Anything from the following that we talked about in class:
      en.wikipedia.org/wiki/IPv4
      http://en.wikipedia.org/wiki/Transmission_Control_Protocol
- Computer component - you, computer, any files in your directory, any files in my directory, no internet. Computer part is second.

Table of Contents

    - Same types of things as hw1, hw2, hw3, quiz3.
    - IPv4 header and TCP header - be able to read them from a file or standard input and pull out parts. Anything we have done in class like this.
    - Practice questions…
        - Read in 5*4 bytes from a file, or stdin, assume it is an IP header, and print off the fields. Note: use fread to read bytes
        - Write out a TCP header into a file, write out raw bytes… Open with fopen(filename, "wb"); and write out with fwrite.
        - I give you a file that we have worked in class, and tell you to make some change… For example, take the gameServer/gameClient and - change the port numbers. Or, something…
        - Or something really simple/basic - do/while loop to read in user input and print it out. for loop to print out 33 -'s.
- 10% is random questions about things you supposedly already know. See https://docs.google.com/document/d/1ObUd5k_iIVWf28LeZYUfBlNjzCLA87zBbM0US5aa5pY/edit#heading=h.70vicaiazljk
    - Simplify log (2^{22}) * 8
    - What is the big-O running time of …. quicksort and insertion sort
    - Hex/binary stuff. What is 92 in binary.
    - 1+2+3+...+101 = ? (101)*(102) / 2
    - 99 + 110 + 121 + … 11*20 = 11 * (9 + 10 + … + 20) = 11 * (1 + 2 + … 20 - (1 + 2 + … 8))
    - $3 + 3^2 + 3^3 + \ldots 3^{12} =$

Grading for Exam 1…

- On paper part
    - Each / is -.5, each X is -1. Except problem 2, for that each / is -.25 and each X is -.5.
    - Is out of 10.5 points.
    - How many points you missed is written on the back.
    - Check my arithmetic and BB.
    - Correct answers, comments in exam1_0.html
        - Note that some problems had different numbers for different people.
- On computer part
    - Question 1 - hex2bin
        - 5/5 if it runs all tests correctly
        - 4/5 if it is basically right just formats the output wrong
        - 3/5 if it is somewhat close...
        - 2/5 if there is it at least does the scanf properly
        - 1/5 if it does something
        - 0/5 if it doesn't compile
    - Question 2 - magicBits

Table of Contents

        - 4/4 if it runs all tests correctly
        - 3/4 if it is basically right just formats the output wrong
        - 2/4 if it is somewhat close...
        - 1/4 if it does something
        - 0/4 if it doesn't compile
    - Question 3 - makeHeader
        - 1.5/1.5 if it runs all tests correctly
        - .5/1.5 if it does something
        - 0/1.5 if you didn't try, or it doesn't compile

In-class quizzes

These will be given roughly once per week, and may or may not be announced. You will generally be asked either a relatively simple question about what was covered in the last class, or you will be asked an extremely simple question about the reading/viewing assignment for that day. The in-class quizzes will count under the category of "homework" in the grading breakdown in the syllabus. Questions on quizzes are normally graded as either correct, incorrect, or half credit. There will be NO makeups for quizzes; if you have an excused absence the quiz simply will not count as anything.

Quiz 3

- Given on 2/12
- Worth 6 points.
- Grading: 3 points part a, 2 points part b, 1 point part c (everything).
- If you didn't have something and finish it before Feb 17, 10am, I'll count it as 80% credit. Normal homework collaboration rules apply.
- Correct working versions called quiz3a, quiz3b, quiz3 in Jeff's usual directory.
- Part a: quiz3a.c
- Part b: quiz3b.c
- Part c (everything): quiz3.c
    - Hint: this does not work:
      unsigned char c; … ; (c << 4) >> 4;
- Finished at 11:24am.

Quiz 2

- Given on 2/10
- Worth 6 points. The better problem is worth 4. The worse problem is worth 2.
- Same as Quiz 1, but also a question like…
- IP/TCP question - If I give you an IP or TCP packet in hex and give you the spec for IP and TCP packet, you can tell me stuff. You can also go the other way - if I give you information, you could write down what the TCP/IP packet would be.

Table of Contents

http://en.wikipedia.org/wiki/IPv4#Header
http://en.wikipedia.org/wiki/Transmission_Control_Protocol#TCP_segment_structure

- The quiz.
  - LAST NAME, FIRST NAME
  - (1)
    - declare an int, scanf the int. call it x;
    - declare a character, scanf a character, call it ch.
    - printf 0 or 1 whether the 3rd bit from the right is a 1 in ch.
    - Example: If ch in binary is 01010b10 you'd output b
      int x; scanf("%i", &x);
      char ch; scan("%c", &ch);
      printf("%i\n", (ch & 0x04) >> 2);
      // printf("%i\n", (ch & 0x04) == 0x04);
  - (2) For the following IP packet, what are the following:
    IP version: 0x0
    Total Length: 0xc07
    Protocol: 0xab
    Destination IP Address: 0x02ab6a85, 0x85.6a.ab.02,
    0x0000: **00**00 **0c**07 **ac**01 **00**25 **90**ab **20**0c 0800 4500 .....%.....E.
  - 0x0010: 02ab 6a85 4000 4006 70be 8b66 0ec9 8b66 ..j.@.@.p..f...f
  - 0x0020: 3774 0050 bb1f eb43 39fe 21b8 6684 8018 7t.P...C9.!.f...
  - 0x0030: 007e 5fa7 0000 0101 080a 5984 b17c 24b1 .~_.....Y..|$.
  - 0x0040: 73bf 6368 2049 5355 3c2f 613e 3c2f 6c69 s.ch.ISU</li
  - 0x0050: 3e0a 3c2f 756c 3e3c 6120 636c 6173 733d >.
<a.class=
  - 0x0060: 2266 6f6f 7465 7253 7562 5061 7265 6e74 "footerSubParent
  - 0x0070: 223e 4953 5520 4c69 6e6b 733c 2f61 3e3c ">ISU.Links<
  - 0x0080: 2f6c 693e 0a0a 3c2f 756c 3e20 2020 2020 /li>.......
  - 0x0090: 2020 203c 2f64 6976 3e0a 2020 3c2f 6469 ... ...</di
  - 0x00a0: 763e 200a 0a20 0a20 3c64 6976 2069 643d v>.....<div.id=
  - 0x00b0: 2266 6f6f 7465 725f 636f 7079 2220 636c "footer_copy".cl
  - 0x00c0: 6173 733d 2263 6f6c 2d72 6967 6874 2073 ass="col-right.s

  Note: I think this information was strange because I had run the tcpdump command with improper arguments. Anyway...

Quiz 1

- Given on 2/5
- Worth 4 points
- Grading - each problem is right, wrong, or half, and the better problem is worth 3 points.
- C code for bit shifting and masks and stuff…
  - LAST NAME, FIRST NAME
  - (1) C code to do…

Table of Contents

        - scanf a 32-bit unsigned integer (2 lines).
          unsigned int x;
          scanf("%d", &x);
        - printf in hex the middle 16 bits.
          printf("%x\n", (x & 0x00ffff00) >> 8);
          printf("%x\n", (x << 8) >> 16);
    - (2) C code to do…
        - scanf a 32-bit unsigned integer (2 lines).
          unsigned int x; scanf("%i", &x);
        - OR the 2nd bit from the left and from the right to make them 1.
          printf("%x\n", x | 0x40000002);
        - printf in hex.

Programming Assignments

HW 7

- Due: April 16
- Late due date: (+1 week)
- Grading: 10 points
    - Grade in ~/handin/project/grade.txt -
    - 10 - good progress ...
    - 7/8 - okayish ...
    - 0 - check back with me by Monday...
- File name: ~/handin/project/README.txt - explain stuff...
- If you want me to tell you what to do…
    - A default is to do a game server and client. You can choose what language to use for both. You can always use C like we've been doing.
    - Server waits for 2 clients, when 2 clients have connected a game starts. Then game server sends message to clients, clients send messages back. The exact set of messages depends on the game.
    - A simple game to start with…
      Goal is to "complete a square". Server sends a random smallish integer, and player needs to enter an integer (positive or negative) that when added to the other makes a perfect square. Player that gets it first wins. One of the players would see…
      Welcome to the Complete the Square Game!
      Waiting for another player….
      Another player has connected. Are you ready? ***yes***
      Complete the square: 10
      6
      That is correct. But, sorry, you were too late. The other player got it first.
      Do you want to play again?

Table of Contents

- Do you want to work with 1 partner?
    - One way is that you both do the same project independently, and just help each other out.
    - Another way is to do one project, but do pair programming, alternating you is typing.
    - Another way is to do one project, split up which functions.
    - Not a good way - you sort of work together and one person ends up doing everything.
- Note: when you apply for jobs, you mention on your resume the projects you have worked on!
- Get a start on your project. Include comment block at top of code describing…
    - What you want it to do in the end.
    - What you have working so far.
    - What I have asked you to do as a starting point, and whether that works or not.
    - If you are working on a team, give a sentence of what roughly each person has done, and a rough guess at how much each person has contributed - both in time and in actual useable code.
- Grading:
    - <= 3/10 - doesn't compile. For a good student, you've spent < 3 hours so far.
    - <= 6/10 - compiles but doesn't remotely do what I suggested. For a good student, you've spent < 5 hours so far.
    - <= 8/10 - sort of okay, but you haven't really spent any time on it. For a good student, you've probably spent at least 5 hours so far.
    - > 8/10 - a nice start. For a good student, you've probably spent at least an hour per day.
    - Note: for time estimates, use an appropriate multiplier. For Jeff, the multiplier is ¼. For people who don't have homework assignments done, the multiplier is maybe 2 or 3.

HW 6

- Due: April 9
- Late due date: (+1 week)
- Grading: 10 points
- File name: myMail.c
- Program to send email using smtp and cs as the smtp server…
- Transcript of running your program…
  to: <you type an email address>
  from: <you type an email address>
  subject: …
  body: …
- Then it sends the email.

Table of Contents

- To figure out how to do it, search for “smtp protocol” and “smtp sample transcript” and “smtp telnet”. Try things out…
- Start by trying to send an email by telneting to cs
  telnet cs 25
  (while you’re already logged in to cs)
- see
  http://www.cs.cf.ac.uk/Dave/PERL/node175.html
- Rule: send a message from yourself to yourself, using a personal email address. If personal email address doesn’t work, use cs473xx@cs.indstate.edu. Note, you can .forward your cs473xx email.

HW 5

- Due: April 7
- Late due date: (+1 week)
- Grading: 10 points
- File name: currentTime.c
- TCP program that is like temperature.c and loads
  http://cs.indstate.edu/~jkinne/cs473-s2015/current_time2.html
  or
  http://www.google.com/search?q=current+time
  or
  use the NIST TIME or DAYTIME protocol - http://tf.nist.gov/tf-cgi/servers.cgi
- And reports the current time.
- You could get it working with
  http://cs.indstate.edu/~jkinne/cs473-s2015/current_time.html
- Correct working program will be there soonish..

HW 4

- Due: March 5 (so, 8am March 6)
- Late due date: (+1 week)
- Grading: not auto-graded by a script….
  - Graded as out of 20 points, but counted as 15 or 26 hw points, whichever is better for you.
  - Points for…
    - Style: 6 pts. comments, indenting, putting things in functions, your code is easy to read, it makes sense, it’s not terribly inefficient, good variable names, don’t have code repeated unnecessarily. 1pt for each of those.
    - Correctness: 11 pts. minimum functionality. 8 pts if it at least workes for 1 or 2 players
    - **Extra functionality: 3 pts. doing something extra.**

Table of Contents

        - **E.g., client in curses. Or pick letters starting at A and … Or Mod the server to do fighting...**
- Take gameServer.c, copy into your directory, change port to your port 73xx
- Take chatClient.c, copy into your directory as gameClient.c, change port to your port 73xx.
- Run them (run server first, then client; stop client first, then server), make sure client is printing messages. The server is setup to run on cs by default, but the clients can run from anywhere.
- Now, make it like Jeff's or better.
- You'll need to…
    - Keep track of nodes with a linked list like the server does. The clients only need to know the row, col, desc.
    - Jeff's code prints a $ for "me", and prints 'A'+desc for the others.
- Jeff will make updates to the protocol probably.
- For the user interface your options are…
    - printf's to reprint the "world" every time there is an update. That is how the gameClient program in Jeff's directory currently works.
    - use curses.
    - Do the client in html/javascript.
    - Do the client in html/java.
- Note on needing to press enter…
    - gameClient.c now has some code so you don't have to press enter after every key. But, only use it if you check for 'q' somewhere in your main loop, so you can setupTerm(0) before closing.

HW 3

- Your file name: hw3.c
- On time due date: Tuesday, Jan 27 by 11:59pm (+ 8 hours)
- Late due date: + 1 week
- Put it in ~/handin/ where you'll put all the code you handin for this class.
- Correct working program: ~jkinne/public_html/cs473-s2015/hw3
- Grading: 10 points, 5/10 points if only first part is done
- Program should…
    - (1) Ask user for IP version, # of bytes to send, create a single 32-bit unsigned integer that has the first "row" (first 32 bits) of an IP header.
        - 4 bits of version, whatever they typed (assume typed <= 15)
        - 4 bits of # of header "rows", always use 6.
        - 8 bits of 0's
        - 16 bits for the number: 6*4 + # of bytes to send
        - You create a 32-bit unsigned integer with all of that, and print it to the screen in hex.
    - (2) Going the other way - inputting a 32-bit integer in hex and pulling out the version, # header rows, # bytes to send.

Table of Contents

- Hint: see bitfun.c for some examples of messing with bits...

HW 2

- Your file name: hw2.c
- On time due date: Wednesday, Jan 21 by 11:59pm (+ 8 hours)
- Late due date: Monday Feb 2 by 11:59pm (+ 8 hours)
- Put it in ~/handin/ where you'll put all the code you handin for this class.
- Correct working program: ~jkinne/public_html/cs473-s2015/hw2
- Grading: 10 points, 5 for the square and 10 if you get square and diamond.
- Program should…
  - Have a scanf to begin main, it scans an integer that you can assume is between 1 and 20.
  - Hints….
    - First goal: print a single line of *'s that is the right length. If integer is 4, then print
      ****
    - Second goal: print the square, that's nested for loops.
    - Third goal: print the square, and print the triangle. To print one line of the triangle is two (not nested) for loops - print spaces and then print *'s.
    - Fourth goal: square and diamond.
  - You display the following if the integer is 4

```
****
****
****
****
   *
  ***
 *****
*******
 *****
  ***
   *
```

  - It should print the right size for integers between 1 and 20

HW 1

- Your file name: hw1.c
- On time due date: Tuesday, Jan 20 by 11:59pm
- Late due date: Monday Feb 2 by 11:59pm (+ 8 hours)
- Put it in ~/handin/ where you'll put all the code you handin for this class.
- Correct working program: ~jkinne/public_html/cs473-s2015/hw1
- Grading: 5 points, all or nothing credit, program must match the correct program exactly.

Table of Contents

- Program should...
- Takes an integer as a command-line argument.
- Run it like: ./myprogram 15
- Print out whether the number is
    - has it's "1 bit" set
    - has it's "2 bit" set
    - has it's "4 bit" set
    - has it's "8 bit' set
    - int x = 15; // print out x & 0x01, or x & 0x02, or x & 0x04, or x & 0x08
- Do it on your own, do not look it up.
- For extra fun, make it do more, include something else after the correct output (e.g., binary representation of the number, count of how many bits are 1, …).
- Hints…
    - int x = atoi(argv[1]); if ((x & 0x01) != 0) ...

HW 0

- Doesn't count for anything but you must do it.
- Login with your cs473xx login.
- Run passwd, change your password to something you'll remember.
- Run chfn, and put in your correct full name, leave other information blank.
- mkdir ~/handin
- echo "jkinne" > ~/myname.txt
  but put in your sycamoreId, not mine
    - Or, edit myname.txt and put in your sycamoreid

Table of Contents

Important Links

Networking

- Exoo's version of this course
- BSD Interprocess Communication I, BSD Interprocess Communication II - Exoo's version of the course followed these for much of the course.
- Computer Networks MIT Course - similar content to what we plan. Has lecture notes, no videos.
- Introduction to Computer Networks at Stanford - has ppt slides for many of the lectures.

C and C++

- How to Think Like a Computer Scientist, C++ Version by Allen B. Downey. Required text, that is an introduction to C and C++. Readings will be assigned from the book.
- CS 50: Intro to CS I at Harvard. We will follow this course. Watching video lectures from the course will be assigned.
- Reference on C and C++. Reference on C and C++ language and standard library. The tutorial on the C++ language is very good. Most of the information up until "Classes" applies also to the C language.
- The C Programming Language - the standard reference for C programming. This is good to use as a reference after you have basic understanding of C and programming (about halfway through the semester).
- C Programming Tutorial
- Codepad.org - run basic C and other code online in a web browser. Note that user input doesn't work (cin, scanf, etc.).
- IdeOne.com - another one to run C code online.
- Fresh2fresh C Tutorial - another C reference/tutorial.

Software and CS Server

- Download Putty. For those using Windows at home, download and install Putty. Then watch the first two videos on this youtube playlist to show you how to use it.
- Getting Started with the CS server - links and videos to help you get started with connecting to the CS server.
- Linux Console Tutorial. A detailed tutorial on how to use the Linux console (command line). Not everything it discusses is available to you (e.g. permissions), but it offers a broad spectrum of what can be done.

Table of Contents

CS Major and Minor

- ISU CS Major Requirements. Scroll down on that page to also see 4 year and 3 year plans of study, and suggestions on non-required courses to take.
- ISU CS Minor Requirements
- ISU CS Prerequisite map
- Contact Information for Math and CS Department

Other Programming

- Scratch Programming - all online, visual, no syntax errors, share with friends!
- MIT Android App Inventor - online, visual, create apps for android.
- Code.org - links to other beginning programming tutorials, many suitable for kids. The organization is pushing teaching programming/CS in K-12 schools.
- Programming at Khan Academy - learn javascript (that is what most interactive websites use)
- Udacity Online Courses - including introduction to CS in Python, intro to Java

Table of Contents

Course Schedule and Notes

Things you hopefully knew before this course

Math

- Arithmetic sum: $1 + 2 + 3 + \ldots + n = n(n+1)/2$
- Exponents:
  - $(b^x)^y = b^{x*y}$
  - $b^x * b^y = b^{x+y}$
  - $b^x / b^y = b^{x-y}$
- Logarithm: $\log_b(a) = x$ means $b^x = a$.
  - For example, $\log_{10}(100) = 2$, $\log_2(16) = 4$. In general, $\log_b(b^x) = x$
  - $\log_b(a^y) = y \log_b(a)$. $\log_2(1000) = \log_2(10^3) = 3 \log_2(10)$
  - $\log_b(a * c) = \log_b(a) + \log_b(c)$
  - $\log_b(a / c) = \log_b(a) - \log_b(c)$
  - Note: we'll almost always use log base 2.
  - Note: log(n) is much smaller than n. Also, $(\log(n))^{100} = o(n^{1/1000})$
- big O means <= up to a constant. So get rid of the constant multiple and it's <=0
- Extra garbage you learn in CS 303. So, information for you but I won't put it on a test.
  - little o means <. So, < eventually no matter what constant.
  - big Omega, Ω, means >= up to a constant.
  - little omega, ω, means >.
  - big Theta, Θ, means = up to a constant.

Algorithms

- Sorting
  - Mergesort, Quicksort, Heapsort - O(n log(n))
  - Insertion sort, selection sort, bubble sort - $O(n^2)$
- Searching
  - Linear search - O(n)
  - Binary search - O(log(n)), only works on already sorted data

Table of Contents

Data Structures

	Insert	Delete	Lookup	Notes...
Unsorted array	put it at the end of array, O(1)	do lookup, O(n) then swap with last, O(1)	linear search, O(n)	easiest to code. declared with some max size.
Sorted array	shift over everything, O(n)	shift over everything, O(n)	binary search, O(log(n))	
Linked list (stack, queue)	update a few pointers at beginning of list, O(1)	do lookup, O(n) then update a few pointers (1)	linear search, O(n)	like an unsorted array, but easy to grow bigger
Binary tree (balanced)	O(log(n))	O(log(n))	O(log(n)), the height of the tree is O(log(n))	but, insert and delete need to do some work to keep it balanced. (Look up AVL tree.)
Binary tree (unbalanced)	O(n) worst case if completely unbalanced	O(n) worst case if completely unbalanced	O(n) worst case if completely unbalanced	
Heap	O(log(n))	O(log(n))	O(log(n))	
Hash table				average worst

Bits and Bytes and Stuff

- binary: 1100 = 8*1 + 4*1 + 2*0 + 1*0 = 2^3*1 + 2^2*1 + 2^1*0 + 2^0*0
- decimal: 1100 = 1000*1 + 100*1 + 10*0 + 1*0 = 10^3*1 + 10^2*1 + 10^1*0 + 10^0*0
- hexadecimal: 1100 = 16^3*1 + 16^2*1 + 16^1*0 + 16^0*0
- decimal: 4567 = 1000*4 + 100*5 + 10*6 + 1*7 = 10^3*4 + 10^2*5 + 10^1*6 + 10^0*7
- hexadecimal: D5F2 = 16^3*13 + 16^2*5 + 16^1*15 + 16^0*2
- why hexadecimal? one hex digit is 4 bits.

Study Guide for this Course...

Terms / Notes

- modem - modulation, demodulation

Table of Contents

- modulation - convert digital bits to analog signals
- demodulation - reverse of modulation
- internet - deals with inter-networking, connecting networks.
- error detection - be able to know if there is an error in transmission
    - methods - parity check bit, checksums
- error correction - be able to correct errors, hopefully without having to retransmit
    - ARQ - automatic repeat request, correct errors by asking to send again if error in transmission
    - FEC - forward error correction, correct errors by using redundancy in what was sent, no retransmission needed
- ACK - acknowledgement
- MAC - media access protocol, protocol for many hosts sharing a single resource (i.e., network wire)
    - CSMA/CD - carrier sense multiple access / collision detection, MAC protocol used in the internet, includes randomized exponential backoff when a collision is detected
- switch - receive data frames and forward them over one or more of its ports
    - circuit switching - setup phase determines who gets the resource when, and then transmission phase makes sure to follow that. no header needed in data. used for telephone networks. bad for bursty, variable-size data. good for constant-rate data.
    - packet switching - data sent in packets with headers that have information about where it should go. good for bursty, variable-size data. internet uses this.
        - datagram routing - header has address of where data should go, switches must have information about where to forward based on the address. IP uses this.
        - source routing - data being sent has a complete route attached to it (after being discovered).
        - virtual circuit routing - hybrid of those two.
        - LAN network - local area network, an ethernet
            - bridge, aka LAN switch - take in data on one link, send it out on other(s)
            - loops cause problems, must be dealt with somehow.
                - distributed spanning tree discovering, transmit only over spanning tree of the switches/bridges.
            - not scalable - each switch must have information about all hosts connected to the LAN, discovery of spanning tree is a linear-time operation (don't want that to happen over the entire Internet).
            - inherently homogeneous - doesn't allow different types of connections.
- router -

Table of Contents

- Internetworking problem - connecting different types of networks - networks with different speeds, protocols, etc. Two issues - dealing with different types of networks, and dealing with a huge number of total nodes (e.g., all computers on Earth)
    - Gateway - interface two or more different networks. Two basic methods -
        - translation - gateway translates packets between different formats for different networks. but scales poorly to large number of nodes and large number of possible types of sub-networks.
        - unified network layer - have standards that all networks must follow
            - Universality - IP-over-everything, Best-effort service model, end-to-end arguments
            - Robustness - soft-state, fate sharing, convservative-transmission / liberal reception.
        - Weaknesses of Internet - assumes trustworthy end hosts, security issues, greedy hosts aren't handled well.

IPv4

- 32 bit addresses
- Class A, Class B, Class C addresses -
    - A - 0, 7 bits of network id, 24 bits of host id
    - B - 10, 14 bits of network id, 16 bits of host id
    - C - 110, 21 bits of network id, 8 bits of host id
    - D - 1110, 28 bits for group address - used for multicasting
    - starting with 1111 reserved for experiments
    - subnet masks - 32 bit mask starting with some number of 1s, then the rest 0s
- MTU - maximum transmission unit. If trying to forward packet to place with smaller MTU than the packet, either
    - discard and send error message back to sender. bit in header says if should do this or not.
    - fragment into smaller packets, let receiver reassemble. the default.
- TTL - time to live, starts at something, decrement by 1 at each router, stop forwarding when reach 0.
- Other IP header fields: TOS (type of service), Protocol (used by higher-up layers), header checksum, IP options (not used much by routers).
- Note… If you were designing IP, you might decide an IP packet looks like…
    - To address - takes 4 bytes
    - Size of packet - takes 4 bytes
    - Other stuff -
    - Data - takes up remaining ___ bytes, for some fixed size that everyone knows about.

TCP

- Transmission Control Protocol
- Mostly done by end hosts, not by routers. Routers do IP

Table of Contents

- Use of IP by routers introduces difficulties: loss of packets, variable packet delays, packets arriving out of order.
- TCP is… **in-order, reliable, duplex, byte-stream** abstraction. For unicast networks (one host to one host)
    - Deal with errors using ARQ. Receiver sends ACK indicating what packet it expects next (meaning it has received all previous ones ok). ACK includes how much space left in its buffer (used for flow control).
    - If loss occurs, then either
        - Timer-driven retransmission - if ACK hasn't been received in certain amount of time, resend.
            - Based on RTT (round trip time)
        - Data-driven retransmission -
    - See also the internet, e.g., http://en.wikipedia.org/wiki/Transmission_Control_Protocol
    -

Socket programming

- Server accepts connections from client(s), doesn't know IP/port of client until client connects.
- Client makes connection to server, client must know IP/port of server.
- Server can have multiple connections on one port.
- C Functions
    - **htons** - converts the uint16_t integer hostshort from host byte order to network byte order
    - **socket** - creates an endpoint for communication and returns a descriptor
    - **connect** - connects the socket referred to by the file descriptor sockfd to the address specified by addr
    - **bind** - assigns the address specified by addr to the socket referred to by the file descriptor sockfd
    - **listen** - marks the socket referred to by sockfd as a passive socket, that is, as a socket that will be used to accept incoming connection requests using accep
    - **accept** - extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket
    - **read** - attempts to read up to count bytes from file descriptor fd into the buffer starting at buf
    - **write** - writes up to count bytes from the buffer pointed buf to the file referred to by the file descriptor fd
    - **select** - allow a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become "ready" for some class of I/O operation
        - Four macros are provided to manipulate the sets. **FD_ZERO**() clears a set. **FD_SET**() and **FD_CLR**() respectively add and remove a given file

Table of Contents

descriptor from a set. **FD_ISSET**() tests to see if a file descriptor is part of the set; this is useful after select() returns.
    - **ioctl** - manipulates the underlying device parameters of special files. In particular, many operating characteristics of character special files (e.g., terminals) may be controlled with ioctl() requests.

Life of a Packet…

- This is probably, mostly true…
- sympodium.indstate.edu wants a webpage from google.com
    - sympodium is 139.102.55.116. google.com is 216.58.216.238
- Firefox/OS get IP address for google.com from a DNS server (domain name server).
    - How does OS know the IP address of a DNS server? It's either set manually using some configuration, or configured using DHCP.
    - And we ask the DNS server for the IP address of google.com
- Firefox wants to get a webpage from 216.58.216.238
- Getting a webpage…
    - use HTTP protocol. Send TCP data with the data part set to
      GET index.html
      Host: google.com
    - Send that packet to 216.58.216.238 on port 80
- Google will respond with TCP data that contains the html code.
- For the actual transfer between my computer and cs.indstate.edu, how does that get routed?
    - First, my computer sends the packet on its network wire to whatever router/switch it is connected to.
    - Router/switch has rules for where to send different IP addresses.
    - One of the basic rules is whether it is a local IP or not. Look at your subnet mask or CIDR address. If my IP & subnet mask == server IP & subnet mask, then we're on the same local network, and switch/router should know what to do.
    - If it's on a local network, which is relatively small, the switch/routers know about all the computers, and do some sort of shortest path calculation to determine where to send packets.
    - If the IP of the server is not on the local network, then we send the request "up the line".
        - How does the ISP router know where to send the address? ISP is connected to some other ISP's which have broadcasted messages telling it which ranges of IP's they have. If the server IP is none those, then we have some uplink to someone who would know.
        - Eventually we'd get to a Tier 1 ISP. The ISP's all basically trust each other about the range of IP's they have.

Protocols

- Link layer -

Table of Contents

    - ARP
- Internet layer - used by routers/switches
    - IPv4/IPv6 - all the traffic is ultimately contained in IP packets.
    - The rest of these are particular cases of IP packets that are sent among routers/switches, and applications normally don't have to worry about them.
    - ICMP
    -
- Transport Layer - used by OS's, applications
    - TCP (over IP)
        - connection-based
        - errors dealt with at network layer, send again, sequence numbers, good for applications where don't want errors/dropped packets
        - used by: HTTP, HTTPS, POP3, SMTP, IMAP, SSH, FTP,
        - http://en.wikipedia.org/wiki/Transmission_Control_Protocol
    - UDP (over IP)
        - connection-less
        - errors handled by application, good for high-speed applications where errors/dropped packets may not matter
        - used by: DHCP, DNS, voice/video
        - http://en.wikipedia.org/wiki/User_Datagram_Protocol

Useful Commands in Linux/Unix

Some also work on Windows, most work on Mac (which is Unix)

- ip link show
  - list network interfaces available
- netstat -i
  - another list of network interfaces, with statistics
- ifconfig -a
  - another list of network interfaces, with configuration information
- ping google.com
  - tests whether can connect to google.com, displays some stats. google.com could choose to ignore ping requests, but most servers don't do that.
- traceroute -A google.com
  - shows information on the route between your computer and google.com
- dig google.com
  - show DNS information about google.com. try for cs.indstate.edu and mathcs.indstate.edu

  dig cs.indstate.edu MX
  - shows the MX record, which is the server to send mail to to get to cs

Table of Contents

- whois indstate.edu
  - shows registration information for indstate.edu. try for other domains.
- tcpdump -nnvXSs 0 -i any port 2150
  tcpdump -nnvXSs 0 -i any host google.com
  - display information on TCP packets sent from your computer to cs.indstate.edu on network interface eth0.
  - see the internet for more info. E.g.,
  https://danielmiessler.com/study/tcpdump/
  http://www.rationallyparanoid.com/articles/tcpdump.html
  - for Windows, use windump - http://www.winpcap.org/windump/
  To run, click Start, type cmd enter, browse to directory windump is in, and run it like windump -nnvXSs 0 -i any port 2150
  - on Mac/Linux, you might need to do
  sudo tcpdump -nnvXSs 0 -i any port 2150
-

What we will do today…

Things will be listed here most recent first.

- Your suggestions and/or later…
  - ARP poisoning
  - man in the middle attack, crypto
  - https, ftps
  - And what all the things in the IP header and TCP header mean.
  - sql injection
- 4/30
  - Project
    - Officially due Friday May 8, 11:59pm
    - You can tell me it's done earlier, and I'll grade it. But I'll only grade it twice.
  - Final exam -
    - Same as exam2, maybe a question or 2 on ARP, DNS. Review the slides linked below.
  - Presentations on ARP, DNS…
    - https://drive.google.com/open?id=1cz1rRieQSJXtJ2kug9QLXoXBO4QN1TdXyxXl1qFmqsc&authuser=0
    - https://drive.google.com/open?id=0ByNqMQF3nAbYMGQ5OWpHbXoycEk&authuser=0
    - https://drive.google.com/file/d/0B0IQwG2s7g6JZ2trcEJZLWk2NzQ/view?usp=sharing
  - Exam2 graded.

Table of Contents

        - Exam 2 c.3 = exam2computer.txt, if I didn't find your file let me know.
            - Note - answers at http://cs.indstate.edu/~jkinne/cs473-s2015/exam2computer.txt
        - Exam2 c.1, Exam2 c.2, Exam2 c.3, Exam2 p, Exam2 computer, Exam 2
        - Exam 2 p = paper, /14
        - Exam 2 computer = max-problem * 13/20 + next-best * … + next-best * ….
        - Exam 2 = 50% computer + 50% paper.
        - Look for an email on where you stand...
        - / is -.5, X is -1. Total # points is 14 for the paper. Number on 3rd page is amount you missed.
        - Computer…
            - reverse - ⅘ - does something with switching around, just not what I asked for, or not quite done. 2/5 - copied bits hw problem and didn't really change it to do what I asked. ⅕ - didn't compile, something there but not right. ⅕ - something that compiles and runs, but hardly does anything.
            - fromThePresident - 3/5 - copied your myMail.c program, and it worked, but didn't make the changes I asked for. ⅗ or 4/5 - looked like you had the right idea, but for some reason did not work. ⅕ - looked like something, but didn't compile.
            - #3 - 5/5 missing 1 piece of info, ⅘ missing 2 items, ⅗ missing 3 or 4 items, ⅖ missing 5 items, ⅕ missing 6 items, 0/5 no correct information.
- 4/23
    - Exam declared done at 12:23pm
- 4/21
    - Exam next time
        - see topics below.
        - on paper - yes
        - on computer - run a few commands, do a socket program.
    - project checkin
    - Things we haven't learned, you don't know…
        - MIT course - L7, L18, others. Wireless networking.
        - Cisco certification?
            - http://en.wikipedia.org/wiki/Cisco_Career_Certifications
            - https://www.google.com/search?client=ubuntu&channel=fs&q=cisco+certification+quiz&ie=utf-8&oe=utf-8
        - Microsoft certification?
            - http://en.wikipedia.org/wiki/Microsoft_Certified_Professional
            - https://www.google.com/search?client=ubuntu&channel=fs&q=microsoft+certification&ie=utf-8&oe=utf-8#channel=fs&q=microsoft+certification+review

Table of Contents

- 4/16
    - Exam …
        - Protocols: TCP (seq, ack,...), IPv4, UDP, SMTP, HTTP,
            - Given a raw packet and a spec, you can say stuff.
            - Given a spec you could construct the packet.
            - Answer basic questions about number of bytes, maximum/minimum length, error checking, etc.
            - IPv4 addresses - CIDR addresses, how many addresses
        - Commands:
            - ping, telnet, tcpdump (windows version), ifconfig, netstat, whois, host, hostname,
            - Do BLANK command and tell me the results, what they mean.
        - C functions:
            - read, write, socket, bind, accept, select, htons, htonl, inet_addr, connect, ioctl
            - memorize the function prototype
        - Basic stuff:
            - binary, hex, math formulae listed above,
            - C programming - files, char arrays, pointers, loops, ...
        - Extra credit or open-ended extra question: ARP, ICMP, RSA, HTTPS, ..., whatever you are learning for your project.
            - Tell me something we didn't do in class.
    - Project …
        - Work on it …
        - I'll look at them tomorrow, maybe.
        - You can finish early and we declare it done. It can be graded early. Graded like gameClient
- 4/14
    - Course evaluations - do them some time.
    - Exam 4/23…
        - Protocols: IPv4, TCP, SMTP, UDP, HTTP
        - On paper questions…
        - Programming questions… I have a server running, and you have to write a program to connect to it and get a secret message. Or something similar to currentTime, myMail.
        - Run networking command questions (find the list of command below)... getting IP address of some website, run telnet to do something like smtp/http, ifconfig to get networking info….
    - All HW's up through myMail due by 4/23
    - The next exam… Let's talk about it.
    - grade currentTime, myMail…
        - show me your currentTime - run it, and code

Table of Contents

        - show me your myMail - run it (send email to jkinne@cs.indstate.edu) and code
        - project?
    - Questions about the project?
    - other commands we should know?
    - UDP and multicast, broadcast…
        - http://en.wikipedia.org/wiki/User_Datagram_Protocol
        - http://tools.ietf.org/html/rfc768
        - http://en.wikipedia.org/wiki/List_of_RFCs
- 4/9
    - currentTime
        - note: if using the google link, what you get in your program is different than you see if you load the page on firefox/chrome/IE and download the source.
        - note: sometimes there is no Content-Length in the header.
        - you need to give the current time.
    - myMail
        - do it in telnet first… telnet cs.indstate.edu 25 and then enter commands. Once you can do that, make the program do it.
    - Project - what is your plan? HW7 ...
    - commands: ping, arp, netstat, ifconfig, host, hostname, route, traceroute, nmap
    - broadcast - broadServer.c, broadClient.c
- 4/7
    - currentTime -
    - myMail - in /u1/junk/cs473
    - Project - rules and the idea….
        - You can use anything, just cite your sources at the top of your code. Ask if you're not sure.
        - Goal is to do some networking something - of your choosing.
        - Some employers like it if you've worked on some nice projects. Then they don't have to train you as much. You list projects on your resume.
        - Ok, we'll do it.
        - Decide for next time what you're going to work on. Get started on it. Maybe HW7 will be due next Tuesday and will be a "project checkin".
    - Project - pick something to do…
        - https - search for something like "openssl c socket program"
        - packet dumper/analayzer - probably have to run it on your own computer, maybe, because of permissions.
        - discover network - run a program that detects other computers that are on the same LAN.
        - grab and print remote file (like our own simple version of ftp) - your own "ftp" server and client. Modify the chatServer, chatClient to transfer a file.

Table of Contents

        - scan CS computers and display information on which ones are up, report back some statistics (e.g., memory, cpu usage, uptime, disk usage, …) - basic information comes from standard linux stuff, but then you have a networked program to compile the information together.
        - ipv6 - make a program uses it. then do something else.
        - bandwidth to CS - in TCP and in UDP, or ipv4 versus ipv6. Kind of like a internet connection speed test (try one out online).
        - Other “measurement” programs.
        - With my approval - try to kill one of the x computers. DoS. Or maybe we’d kill something running in a virtual machine. We’d get Steve Baker’s help.
        - Play a midi over multiple machines - have them synchronize. First look how to play a sound (use the beep).
        - Anything else we haven’t done that you wanted to do?
        - “no programming project” - pick a lecture from the MIT course we didn’t do, and learn it well, and present it to the class well.
    - UDP and multicast, broadcast…
        - http://en.wikipedia.org/wiki/User_Datagram_Protocol
        - http://tools.ietf.org/html/rfc768
        - http://en.wikipedia.org/wiki/List_of_RFCs
        -
- 4/2
    - HW5 and HW6 - problems, due dates, points
    - overview of encryption… asymmetric, symmetric, RSA, AES,
        - Reading…
            - http://en.wikipedia.org/wiki/RSA_%28cryptosystem%29
    - telnet’ing to look at http, smtp, ftp protocols.
        - try http://www.cs.cf.ac.uk/Dave/PERL/node175.html.
        - Don’t do bad things, or else we can’t teach this class anymore, or our server gets shut down (and your account gets removed, and you fail the class).
- 3/31
    - 2’s complement - remember?
    - Next exam date? Second exam is April 23rd, final exam is during finals week and basically the same material.
    - gameClient? have it done for real in a week.
    - tcp/http example...
    - Next hw assignments… http, ftp, dump, ping
    - Project?
    - udp - broadcast, multicast
        - http://www.tack.ch/multicast/broadcast.shtml
- 3/26
    - Fields Metal

Table of Contents

    - finish grading gameClient...
    - MIT lecture 4
    - Know the acronym stuff…
    - The next assignments and/or in-class examples will be…
        - something like ping
        - something using either http or ftp. not a full-fledged client, but something that interacts with a server. For example, gettime program that checks time.gov. Then end of TCP. http is port 80, ftp is port 21. So basically open a socket to the right port, then send the appropriate message and wait for response.
        - some UDP stuff and/or packet sniffer stuff.
    -
- 3/24
    - Acronyms: IP, TCP, ARP (address resolution), BGP (border gateway), iBGP, eBGP, AS (autonomous system), RIP, OSPF, ISP (Tier1, Tier2, Tier3), MIT lecture 4 page 12 table 1, MTU, MSS, SEQ, ACK, SYN, FIN, cksum, DF
    - /usr/sbin/tcpdump -r dumped_tcp.txt
      /usr/sbin/tcpdump -vvv -X -r dumped_tcp.txt > ~/delme
    - Read: http://en.wikipedia.org/wiki/Transmission_Control_Protocol
    - Review the slides from last time. And MIT lecture 4
    - New HW/project:
      ftp - we do an ftp server and client
      http - simple website reader
      bitmap?
      using encryption/decryption?
    - encryption, public-key, private-key, etc.
        - symmetric encryption - encryption/decryption are the same for sender and receiver. Examples: caesar cipher, all encryption before 1970. Currently: AES, DES. Same password/key for both sides of communication.
        - asymmetric encryption - public-key is an example. Different password/key for encryption than for decryption. Examples: RSA, others.
- 3/12
    - Questions from last time? - public/private key, etc.
    - gameClient - yep.
    - Next assignment / in-class example?
    - Details of TCP…
    - tcpdump, hostname, dig, whois, ping … tutorial from MIT course.
- 3/10
    - Guest lecture from Andrew Chi. Slides: https://drive.google.com/file/d/0B7xM6icc2fOUckhLajhzcnVmZUk/view?usp=sharing
    - http://bgp.he.net/
    - Coming up: UDP and other stuff. Take a look at /u1/junk/cs473/apr14/icmp.c

Table of Contents

- 3/5
    - Note about total grade and letter grade so far.
    - HW4 game - I'll look at it this weekend, due today. Remember that 15% of the grade for it is "something additional". You should put a comment at the top of your code saying what this is. Also remember that part of your grade is "good programming style".
        - Things people have done extra:
            - curses UI
            - fighting/points/health. zombies versus humans.
            - auto display people from a to z
            - getting IP address off computer auto.
    - dealing with raw files, bitmap example.
- 3/3
    - Go over the exam…
        - Remember that this exam gets dropped if you do better on the next. final exam grade will be
          max(avg(exam 1, exam 2, exam 3), avg(exam 2, exam 3), exam 3)
        - Computer test - remember to make sure your program compiles!
    - thrashing - when the programs you are running take more memory than you have, the OS tries to use the hard drive as memory (called virtual memory), but things keep getting sent back and forth between memory and hard drive, and none of the programs ends up being able to do what it needs to. For you, if you constantly are looking up what functions do, can you get anything done? The answer - spend time doing the programs, and you'll start to remember.
    - Game due tonight.
- 2/26
    - Interim grades -
    - Behind on grading, plan to catch up this weekend.
    - Go over the exam? Next time.
    - gameClient due tonight? Next time.
        - killing a rogue process…
        - top to get the pid. or ps -aux.
        - then kill pid, if that doesn't work kill -9 pid.
    - Next up - UDP, other kinds of IP traffic. Packet sniffer program.
- 2/17
    - quiz3 graded
        - Remember on hw's and quiz's like this run
          ~jkinne/public_html/cs473-s2015/testProgram.sh quiz3
    - test on Thursday
    - gameServer -
        - Max number of possible connections = 26, players get alpha letters
        - Players are either "attacking" or "defending". While attacking, if kill a defending player get a point.

Table of Contents

        - Possible messages to send to do this
          attack [desc] 0 0
          defend [desc] 0 0
          points [desc] [numPoints] 0
        - Attacking players should be upper-case. Defending players should be lower case. Players should initially be defending.
        - Player points should be initialized to 0.
- 2/12
    - Quiz today? yes indeed.
    - Let's repeat the same kind of thing next time. And you won't have a chance for late credit.
    - Exam in 1 week, Feb 19? Yes. Look in table of contents for what we'll do. Complain next time if you see any problems with the outline.
    - example with ncurses - mazeGame2.c
        - for more on curses, http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/
- 2/10
    - Quizzes - if you have a legitimate reason to miss class I'll do a makeup or not count the quiz. If you don't have a legitimate reason, it's a 0. Be honest and honorable.
    - Quiz 1 returned
    - Quiz 2 - yep.
    - For the foreseeable future, plan on a quiz every day. We may do them on the computer.
    - example with strtok
    - gameServer
        - RemoveBadConnections
- 2/5
    - We'll end at noon today. Because of Sternfeld. Thank him.
    - Quiz - see above.
    - Guaranteed for sure, next time hw1, hw2, hw3 are done.
    - client and server programs, HW4
- 2/3
    - Note that your port number is the last 4 digits of your cs473xx login.
    - hw1, hw2, hw3 regraded again.
        - hw1, hw2 - ready for the answers… yes, they are done.
    - hw3 - use an unsigned int, not an int to store the information in the first part. why?
    - chatServer, list of client connections - they code keeps a list of client connections in a linked list. Why not keep them in an array or vector?
    - We want to do something with the client/server programs - make them better/safer, and/or make a game or something out of them. Make them have logins, names, avatars. file transfer? detect which other "servers" are running?

Table of Contents

- **Quiz for next time…**
  - Counts in HW grade, 4 points.
  - C statements to grab bits out of integers and put them back in, etc. Be masters of & | ^ << >>.
  - C statement to scanf an integer and print whether the 3rd bit is 1.
    - if (x & 0x20000000 != 0) printf("1"); else printf("0");
  - C statement to take 4 unsigned chars and put them into an integer as the 4 bytes of the integer.
    - unsigned char vals[4] = {0xff, 0x0f, 0xea, 0x28};
    - unsigned int x = ((int) vals[0] << 24) | ((int) vals[1] << 16) | ((int) vals[2] << 8) | (int) vals[3];
  - Not for credit yet: question about IP and TCP packets/headers
- We might have an exam Feb 17 or 19
  - Content: ???
- tcpdump - see notes above on running it
  - Example from running on CS and seeing a packet that came in from client when client typed hi

11:41:39.137730 IP (tos 0x0, ttl 64, id 18670, offset 0, flags [DF], proto TCP (6), length 54) 139.102.14.201.2150 > 139.102.55.135.44555: Flags [P.], cksum 0x5d45 (incorrect -> 0x1a68), seq 2923622997:2923622999, ack 1231965582, win 114, options [nop,nop,TS val 898900809 ecr 2696916655], length 2

0x0000: 4500 0036 48ee 4000 4006 *94b7* 8b66 0ec9 E..6H.@.@....f..
0x0010: *8b66 3787* **0866** ***ae0b*** **ae42 f255** ***496e 4d8e*** **.f7..f...B.UInM.**
0x0020: 8018 0072 ***5d45 0000*** **0101 080a** ***3594 2349*** **...r]E.....5.#I**
0x0030: a0bf aeaf 6869 **....hi**

  -
- hw4?
- first exam - when, what?
- Read -
  - http://docs.freebsd.org/44doc/psd/20.ipctut/paper.pdf
  - http://docs.freebsd.org/44doc/psd/21.ipc/paper.pdf

- 1/29
  - Note on grading - I won't have emails sent to you anymore. The testProgram.sh script is good enough now that I'm using it for grading as well. So you can run it yourself to see what it says. I updated it so that it normally doesn't care about whitespace (except hw2), and it ignores any extra output that is *after* the correct output. Also, your programs must compile with g++ on the CS server.
  - So, hw1, hw2, hw3 - one last chance?
  - Playing with Exoo's client/server programs, in /u1/junk/cs473/
    - See http://www.linuxhowtos.org/C_C++/socket.htm
- 1/27

Table of Contents

    - testProgram.sh in ~jkinne/public_html/cs473-s2015 and using it to check your program.
        - Note that right now it checks for an exact match, including whitespace. For hw1, you could have extra stuff at the end and that's still fine. If the program says your hw1 is wrong because of that I'll check it myself.
        - cd ~jkinne/public_html/cs473-s2015/
        - ./testProgram.sh hw1
            - Makes sure the following files all exist:
                - ~jkinne/public_html/cs473-s2015/hw1.test.txt
                    - test cases for the program to try
                - ~jkinne/public_html/cs473-s2015/hw1
                    - correct working program
                - ~cs473xx/handin/hw1.c
                    - your version.
            - If those all exist then try to compile your program with g++
                - g++ ~cs473xx/handin/hw1.c -o ~cs473xx/handin/hw1.compiled
            - If it compiled, then run the tests from the test file, and print out any test cases where the output was not an exact match.
    - The plan - start socket programming. Let's start with…
        - http://www.tutorialspoint.com/unix_sockets/index.htm
- 1/22
    - hw1, hw2 graded in blackboard. will regrade sometime for late credit (80%).
    - your code MUST….
        - use whitespace appropriately - every time there is a { then the next line is spaced over a little bit more. Also use newlines sometimes.
        - If you don't know what proper indenting should look like, look at mine or also use
          indent myprog.c
        - some comments.
        - reasonable variable and function names.
    - if I happen to look at your code and it does not use good style, I will take off points.
    - hw3?
    - "useful commands" above, what each one means-ish…
    - review of working with files in "binary mode" - example?
    - TCP/IP - terms, and stuff...
- 1/20
    - myname.txt should have your sycamore id (the part of your sycamores.indstate.edu email address before the @).
    - Demo hw2. hw1, hw2 due tonight, tomorrow.
    - Continuing L1, L2, L3, T1.
    - Near-term goals…

Table of Contents

        - IP packet - what does it actually look like.
            - Example, you want to send message “Hello There” in an IP packet, what actually gets sent?
            - Going the other way as well, I will give you an IP packet and you need to “decode” it.
        - TCP packets - same thing - real examples.
        - And ultimately, you don’t understand TCP/IP unless you could write the code for them.
            - So we should make some pseudocode for what the TCP/IP handlers look like.
- 1/15
    - Demo hw1, let people ask questions about their programs if they’re having problems, or if they’re having problems logging in.
    - Say something more about the stuff above you supposedly know from previous courses.
    - Getting started on L1 in the MIT course - some notes are above.
    - Probably a HW2 to keep people practicing their programming - based on what we do in class...
- 1/13
    - Went over the syllabus and basic information.
    - Handed out cs473xx logins to the CS computers that you’ll use for this course. Come see me if you need one.
    - Make sure to login to BB at the beginning of each class and do the attendance quiz that logs your attendance.
    - Started putting in notes for “things you should know/remember” from previous courses above. These are things that come up over and over again in the upper-level CS courses, so it’s worth reviewing. And I’ll include a section on each test asking basic questions about these things, even though it’s material from another course. These things will come up in this course too, and putting them on the test will give you motivation to study them. For anything that is unfamiliar, look it up again - in whatever source you had in a previous course, on wikipedia, in youtube lecture videos, etc.
    - If you’ve never logged in to the CS computers before, check out http://mathcs.indstate.edu/dept/academic/programming.php and the youtube videos linked from there.
    - See assignments (click on the table of contents to find them in this document) for HW 0 that you should do ASAP, and HW 1 that you should work on before next class. We’ll be doing C programming in this class, so I’ll have some practice programs at the beginning of the course that will count for some points.
    - Also, look at the reading assignments and read through them for the next class.
-

Table of Contents

Table of Contents

Email Log

If I remember, I'll copy/paste emails to the class here so you can refer back to them in case you accidentally deleted one.

- 2/18/2015
    - Some people asked for examples of programs that read and write IP and TCP headers. I put two working programs out there that you can try, and then try to write you own program that behaves the same. If people want I could put my source code where you can see it later tonight. What's out there...
    - readIP - a program that reads a binary file that has an IP packet and prints out all the header fields. ipPacket.dat is an example binary file that has an IP packet in it. If you want to view the ipPacket.dat file to see what is in it, you need to use a hex editor. One thing you could do is read in 5 unsigned integers, convert the byte order to what we are used to with ntohl (you'll need to #include something to use that) and then use your bit masks and shifts to get the individual fields.
    - writeTCP - a program that takes command line arguments. The first argument is an input data file, you can use anything. The second argument is the name of a file to save the TCP header and data into. The next two arguments are a source port and destination port. The program then writes the TCP header and data into the output file. Note that it skips writing the IP header, so you can see just what the TCP header looks like. Again, you'll need a hex editor to view the output.
    - That's it. Have fun...
- 1/16
    - Note that the hint had a mistake in it, because of the way C does order of operations. Instead of (x & 0x02 != 0), you should use ((x & 0x02) != 0). Interesting.
- 1/13
    - Note that I updated the google doc for the course (go to kinnejeff.com and click the CS 473 link) with information about what we did today and what we'll do next time. I also updated some information about HW 1. Check the table of contents in the google doc to find all that.

Table of Contents