Announcements (0:00 – 7:00)

Race Conditions (7:00 – 11:30)

- Consider ATM with following problem. Wants to dispense money to you and update database, but needs that these two things happen at the same time.
- Otherwise, if a malfunction occurs in between the two steps, there will be a discrepancy between what the database says and what should actually be in your account.
- The operation needs to be *atomic*, that is, we need that either both things happen or neither does.
- This is difficult because computers can only do one thing at once, even though they sometimes give us the illusion of doing multiple things by rapidly switching between them.
- This won't be a huge concern in this week's problem set, but you might have to deal with these issues in the final project.
- One useful thing in MySQL is the notion of a transaction. All the stuff contained in the transaction is guaranteed to happen at the same time.

Demonstration of Problem Set 7 (11:30 – 14:00)

Implementing a Dictionary in PHP (14:00 – 25:00)

- See speller.php, written by David before class.
- Home exercise: convince yourself that this does the exact same thing as speller.c.
- Notice that speller.php includes dictionary.php. See dictionary.php.
- Size is stored in variable $size and dictionary is stored in an associative array
- The associative array data structure in PHP is a mapping between keys and values
- In load, we do the same stuff you probably did in your C version
    - Open the dictionary file using fopen
    - Read in a word using fscanf
    - Put the word in the dictionary
    - Increment the size variable
    - Repeat
    - Close the dictionary file when all done
- The difference, besides syntax, is that to store the word in $dictionary, we enter the key-value pair ($word, TRUE) into it. This is done by simply setting $dictionary[$word] = TRUE
- To check, all we do is see if there is an entry in $dictionary for $word, and if the corresponding value is TRUE. If it is, we know that we've entered the word, so we return TRUE. Otherwise return FALSE.
- The reason this is so simple is that PHP, as a high level language, already has this sophisticated data structure called an associative array implemented. Someone

else has gone through all the details of figuring out what's most time and space efficient, and we can just use the interface they've created.
- Unload is also trivial because PHP has built in memory management. We don't have to malloc and free explicitly.
- PHP is an interpretive language, which means that it is read in and executed line by line by an interpreter, rather than compiled into an executable and then run.
- The top line of speller.php (beginning with #!usr/) tells the computer what program to use to interpret it
- This implementation is pretty slow: 2.75 seconds (as compared is 0.6 second for the staff solution in C).
- We could try to make some optimizations, but PHP does not give us control over low level details as C does, so it would be difficult to get that time very low.
- Here we see the tradeoff between simplicity of implementation and performance

RSS (25:00 – 33:30)

- A file format that is XML
- An RSS feed is a big text file that contains a bunch of information that you might want to syndicate
- For example, nytimes.com syndicates their news in an RSS feed, so you can just get the RSS feed containing the text, and integrate it into your website
- In problem set 7, RSS feeds are used to get stock quotes and news articles from Yahoo! and MSN
- RSS feeds can help you to easily use publicly available data
- Ppublicly available data and publicly available APIs are a powerful combination, as you'll see in problem set 8!

Introduction to Javascript (33:30 – 41:00)

- Javascript is interpreted, but, in contrast to PHP, gets interpreted by clients not servers
- Recall that PHP is interpreted on server to generate HTML, which is sent to user's browser
- Javascript is "shipped raw" to user's computer, and user's browser will interpret it
- You can embed javascript in your webpage using <script> … </script> tags in the head, with attribute type = "text/javascript"
- You can put javascript right in your html file, or you can put the javascript in a separate file and refer to it using the src attribute
- See slide 11. This snippet of javascipt code puts the cursor in the form field where we want focus to be by default

A Simple Form in Javascript (41:00 – 54:00)

- document.forms gives you access to all of the forms in the webpage

- document.forms.login.username refers to the username field in the login form.
- See form1.html. This contains a form with username and password fields. Its action is process.php
- What happens if we try submitting an empty form? We simply get a printout of everything that was submitted. This is because process.php contains the line "print $_REQUEST"
- $_REQUEST refers to everything that was submitted, either $_GET or $_POST
- We would like to have some error-checking
- In form2.html, there is some client side validation.
- This is done by means of a javascript function called validate() that checks each element of document.forms.registration is ""
- If any is "" it shows an alert with an error message
- Additionally, it checks to make sure the two password fields match, and shows an alert if they do not
- This is called by setting the onsubmit attribute of the form to be "return validate()"
- Problem remaining: a malicious user could bypass this validation by saving a local copy of form2.html to their computer, removing the validation, changing the action to have the full address of process.php, and now entering whatever they want

More Sophisticated Validation (54:00 – 65:00)

- Javascript is an object oriented programming language
- An object is like a struct, in that it encapsulates a set of data members, but, unlike a struct, it also encapsulates a set of functions
- In form3.html, we pass the validation function the form object using the "this" keyword
- This allows us to avoid typing document.forms.registration in every single branch of the if-else
- Instead, we can simply refer to f, the argument passed to the function
- In form4.html, we prevent the Submit button from being depressable so long as the terms and conditions box is unchecked
- The box is initially disabled by setting the disabled attribute to be "disabled" on the input object with name = "button"
- Its value is toggled by setting the onclick event of the checkbox to be the toggle() function. This means that when the checkbox is clicked, toggle() will be called. This function inverts the value of the disabled property
- In form5.html, the input for the email address is checked to make sure it ends in .edu
- This is done by checking that the value of the email field matches the regular expression .+@.+\.edu$
- See slide 13 for regular expressions reference

- We perform the check using the match() method of string class, to which document.forms.registration.email.value belongs

Global Objects, Objects, and Arrays (65:00 – 75:00)

- Javascript has a bunch of global objects, which are data types. Each has some associated methods already written for you. See slide 14.
- An object in javascript is a container that allows you to associate keys with values. It is like an associative array in PHP.
- You can use this framework to associate variables with objects, functions with their implementations
- You can put a key-value pair in an object by saying obj.key = value; OR obj[“key”] = value;
- You can also initialize an object with key value pairs by saying var obj = { key : value};
- Arrays in PHP and javascript are automatically resized: if you put something in an index that doesn’t already exist, the array will grow to the desired size to accommodate that action.
- See slide 20. Here David uses a simple javascript function to reproduce the blinking effect, since the <blink> tag in HTML is no longer recognized. This function simply toggles the visibility of all elements in the document named “blink”