week8

Tommy MacWilliam

tmacwilliam@cs50.net

October 31, 2011

Announcements

- pset5: returned
- final project pre-proposals due Monday 11/7
  - http://cs50.net/projects/project.pdf
- CS50 seminars: http://wiki.cs50.net/Seminars

Today

- common pset5 mistakes
- HTML
- chmod
- CSS
- PHP
- SQL

Hacker Tip of the Week

- tired of typing `cd ~/cs50/psets/pset5/bmp` all the time?
- `ln -s /path/to/target name`
  - create a link to a folder!
  - `cd` into that link, and you'll be in the folder

#1 Design Mistake

- `malloc`: with great power comes great responsibility
- always always always check for `NULL` when you `malloc`
  - always.
  - seriously.
- without a check for `NULL`, your program could segfault if something goes wrong with `malloc`
  - then your entire program crashes
  - probably not the best way to respond to an error

#1 Design Mistake

- `malloc`: with great power comes great responsibility
- don't `malloc` if you don't have to!
  - if what you're `malloc`-ing is only being accessed in the current function, then just put it on the stack

```
int n = 5;
int stack_array[n];
int* heap_array = (int*)malloc(n * sizec
```

  - C allows for variable-length array declarations without `malloc`

HTML

Tommy
MacWilliam

- **H**yper**T**ext **M**arkup **L**anguage
- used to describe the structure of a document
  - aka, a web page
- **NOT** a programming language
  - HTML **describes** content, but does not **create** content
  - no ifs, fors, and whiles
- tells your web browser (Chrome, Firefox, Safari, IE) how to display the content on a web page

HTML

- HTML consists of a series of embedded elements
  - an element is a single “thing” on a web page
- an element is defined by a tag
  - syntax: `<tag>this is in the tag!</tag>`
  - a tag must have a start and an end
  - everything in between the start and end tag is the content of the tag
- good style: always close your tags!

HTML

- common HTML tags
  - `a`: “achor” aka hyperlink
  - `h1, h2, ... h6`: headings (h1 like a title, h2 like a subtitle, etc.)
  - `p`: paragraph
  - `b, i, u`: bold, italics, underline
  - `ul, ol, li`: bulleted/numbered lists
  - `table, tr, td`: tables
  - `img`: image
  - `br`: line break
  - `div, span`: page divisions (for layout)
  - `form, input`: form elements for text input

HTML

- that's not all, a valid HTML document also has a particular structure
- doctype: tells the browser what type of document this is
  - HTML5 doctype: `<!doctype html>`
  - has no closing tag–simply a declaration of the type of file to be used
- `<html>`: tells the browser that everything inside will be HTML
- `<head>`: tells the browser what to load before the user can interact with the page
  - JavaScript files, CSS stylesheets, page title, etc.
  - no content in the head will actually be displayed to the user
- `<body>`: the actual content of the page

HTML

Tommy
MacWilliam

- example time!
  - skeleton.html

HTML

- many tags contain attributes, which modify or give specificity to a tag
  - example: I have a hyperlink, but where do I want the page to link to?
- attributes are specified in key value pairs
  - key="value"
  - example: `<a href="http://cs50.net">CS50 home page</a>`

HTML

- example time!
  - page.html
- to view your page from a web browser, it must be in a directory called `public_html` (or a subdirectory thereof)
- as a CS50 cloud user, you have a URL at http://cloud.cs50.net/~username

chmod

Tommy MacWilliam

- Forbidden: I don't have permission to access?
- on the could, each file has different permissions (read/write/execute)
  - we need to make sure that only certain users are allowed to edit certain files
  - for us, the "words" file in pset6 was "read-only" so we couldn't edit it
- why are permissions important on a web server?

chmod

- without permissions, we could allow users to write and execute arbitrary code on our server!
  - aka being hacked
- by default, all files on the cloud are only readable/writeable by the user who created them
  - so if I create a file on my account and you try to `cd` to my account and edit it, you'll get a permissions error
- in order for a user to see our file from a web browser, we need to grant the user permissions

chmod

- the `chmod` command allows us to manually specify the permissions for a given file
  - syntax: `chmod <permissions> <file>`
- permissions given as an octal number
  - each digit represents a different combination of read/write/execute
  - 3 digits: one for you, one for your group (don't worry about groups), one for the rest of the world

chmod

- chmod cheat sheet: what permissions does each digit correspond to?
  - 7: read, write, execute
  - 6: read, write
  - 5: read, execute
  - 4: read
  - 3: write, execute
  - 2: write
  - 1: execute
  - 0: no permissions

chmod

- chmod 600: user can read/write, rest of the world has no permissions
  - PHP files
- chmod 644: user can read/write, rest of the world can read
  - HTML, JavaScript, CSS, and image
- chmod 711: user can read/write/execute, rest of the world can execute
  - directories containing public web files

CSS

- **C**ascading **S**tyle **S**heets
- HTML defines the content and structure of our documents, while CSS defines their style
  - how they look aesthetically: size, color, etc.
- syntax: `<selector> { <attribute>: <value>; }`

CSS

- CSS selectors provide a way to give style properties to a specific element or group of elements
- to style a specific element, give it an `id` and/or `class` attribute
  - example: `<img src="logo.png" id="logo" class="top" />`

CSS

- common CSS selectors
  - `#<id>`: select the element with the given id
  - `.<class>`: select all elements with the given class
  - `<tag>`: select all elements with the given tag
  - `*`: wildcard, select all elements

CSS

- combining CSS selectors
  - `tag.class`: select the elements of the given tag with the given class
  - `.class1, .class2`: select all elements with class “class1” or “class2”
  - `#id <tag>` / `.class <tag>`: select the descendents of the id/class with the given tag

CSS

- common CSS attributes
  - `color`: color of an element
  - `background`: background color/image of an element
  - `width / height`: size of element
  - `margin / padding`: space between an element and its containing element
  - `font`: font size, family, etc.

CSS

- 3 ways to style your page with CSS
  - create an external stylesheet, then use the `<link>` tag to add it to the page
    - example: `<link rel="stylesheet" type="text/css" href="style.css" />`
  - use the `<style>` tag
    - example: `<style> a { color: red; } </style>`
  - use the `style` attribute
    - example: `<p style="width: 300px;">`

CSS

Tommy
MacWilliam

- example time!
  - pagecss.html, pagecss.css

PHP

Tommy MacWilliam

- PHP: PHP Hypertext Processor
  - recursive acronyms ftw!
- web scripting language used to create dynamic web pages
  - static web page: content is the same every time you view it
  - dynamic page: page content depends on user actions, etc.
- interleaves with HTML pretty nicely

PHP

- all PHP code must be inside the `<?php ?>` tag
- syntax very similar to C
  - `if (condition) {} else if (condition) {} else {}`
  - `while (condition) {}`
  - `for (<pre>; <condition>; <post>) {}`

PHP

- PHP does, however, have small syntactic differences
  - all variable names must start with a dollar sign (e.g. `$x = 5;`)
    - this will get very annoying, I promise
  - functions/variables are not explicitly given types at declaration
    - example: `function increment($x) { return $x + 1 }`

PHP

- arrays in PHP do not have a fixed size
  - add an element to the end of an array: `array_push`
  - get value of last element and remove it from array: `array_pop`
  - get value of the first element and remove it from the list: `array_shift`
- how can we implement stacks/queues with PHP arrays?

PHP

- arrays in PHP are also ordered maps (like a hashtable)
- an integer/string key has an associated value
  - an array can contain keys/values of different types
  - example: `$person = array("name" => "tommy", "job" => "TF", "rank" => 1);`

PHP

- example time!
  - arrays.php

PHP

- PHP code can occur anywhere in an HTML document
  - all PHP code on a page will be executed on the server, then the resulting page will be displayed to the user
- PHP and HTML can be inter-mixed (a lot nicer than using the `echo` function to write HTML)

```
<?php if ($x == 5): ?>
<p>x is 5!</p>
<?php else: ?>
<p>x is something else!</p>
<?php endif; ?>
```

PHP

- a class is just like a struct
  - contains multiple fields of different types
  - unlike a struct, a class can contain functions
- an instance of a class is called an object
  - every class has a special variable called `$this` that refers to the current instance of the object
- fields of an object are accessed using `->`
  - fields do not include the dollar sign (phew!)

PHP

Tommy
MacWilliam

- example time!
  - classes.php

PHP

Tommy MacWilliam

- when your browser makes a request to a web page, it either makes a GET or POST HTTP request
  - GET: request the content of a web page
  - POST: send data to the web page so the server can do something with it
- both GET and POST requests can pass arguments to a PHP script
  - GET variables specified right in the url
  - POST variables hidden to the user, sent in POSTDATA field of HTTP request

PHP

- variables sent to PHP script in key/value pairs
  - PHP can access these variables with the built-in `$_GET` and `$_POST` arrays
- if user accesses the url:
  `file.php?name=tommy&job=TF`
  - in `file.php`: `$_GET["name"] == "tommy";`
- if user submits a form, variables can also be stored in either array
  - form's `action` attribute determines what script variables should be sent to
  - form's `method` attribute determines if POST or GET request should be used
  - input's `name` attribute is key, user's input is value

SQL

- **S**tructured **Q**uery **L**anguage
  - pronounced “see-kell,” saying “ess-kew-ell” just sounds silly
- allows the programmer to interact with a database
  - insert new values, retrieve values matching certain parameters, and delete values
- SQL and PHP are also tightly integrated

SQL

Tommy
MacWilliam

- a database is a collection of tables
  - a table describes a single type
  - example: a “user” would be described by a table
- a table is a collection of fields (columns)
  - example: a “users” table would have columns for username, password, email, etc.
- a row in a table is a single instance of something
  - one user would occupy one row in the users table

SQL

- insert new row: `INSERT INTO <table> (<column 1>, <column 2>) VALUES (<value 1>, <value 2>)`
  - example: `INSERT INTO users (username, password) VALUES ('tommy', 'supersecret')`
- delete row: `DELETE FROM users WHERE <column> = <value>`
  - example: `DELETE FROM users WHERE username='djm'`

SQL

- modify values in an already existing row: `UPDATE <table> SET <column> = <value> WHERE <other column> = <value>`
  - example: `UPDATE users SET password='evenmoresecret' WHERE username='tommy'`
- retrieve values: `SELECT <column> FROM <table> WHERE <other column> = <value>`
  - example: `SELECT password FROM users WHERE username='tommy'`

SQL

- SQL statements executed via `mysql_query`
- phpMyAdmin is a web interface for creating/viewing tables
  - accessible at: `http://192.168.56.50/phpMyAdmin`
- a database for pset7 should already be created for you
  - now you can add new tables and columns to tables

Practice Problems

Tommy MacWilliam

- make a web page that displays two 200 pixel squares (one black, one orange) side by side
- display all the GET variables passed to a PHP script
- create a form that asks the user for his/her name, then displays it