LING 408/508: Programming for Linguists

Lecture 9
September 28th
Today's Topics

• expect
• Web programming
  – html/javascript
Virtual Box

• A note on using copy/paste within and outside to the host system
Control+Shift+C

Control C = (^c) interrupt signal

enable copy from virtual machine into host machine: Devices > Shared Clipboard > Guest to Host
Interaction using expect

- expect is a program that executes preprogrammed interaction using commands including:
  1. `expect string`  
     * look for string in input
  2. `send string`  
     * respond with string (/r for return)
  3. `spawn string`  
     * execute a program under expect
  4. `interact`  
     * gives control to user
     * output will appear on terminal

```
#!/usr/bin/expect -f
expect "hello"
send "world\n"
expect is written using a programming language called TCL (Tool Control Language) aka "tickle"
```
Interaction using `expect`

- Knock knock joke responder:
  1. `#!/usr/bin/expect -f`
  2. `expect "knock knock\n"
  3. `send "who's there?\n"
  4. `expect -re "(.*)\n"
  5. `send "$\expect_out(1,string) who? \n"
  6. `expect "\n"
  7. `send "Very funny\n"`
Interaction using expect

• Expect timeout:
  – "The default timeout period is 10 seconds but may be set, for example to 30, by the command set timeout 30.
  – An infinite timeout may be designated by the value -1."
Interaction using expect

• expect can pattern match against regular expressions using flag –re
  • expect -re "(.*)\n"

• matching patterns are stored in variables:
  – $expect_out(1,string)

Upon matching a pattern (or eof or full_buffer), any matching and previously unmatched output is saved in the variable expect_out(buffer). Up to 9 regexp substring matches are saved in the variables expect_out(1,string) through expect_out(9,string). If the -indices flag is used before a pattern, the starting and ending indices (in a form suitable for lrange) of the 10 strings are stored in the variables expect_out(X,start) and expect_out(X,end) where X is a digit, corresponds to the substring position in the buffer. 0 refers to strings which matched the entire pattern and is generated for glob patterns as well as regexp patterns.
Interaction using expect

• Let's interact with the BMI homework solution ...

#!/bin/bash
# usage: weight height kg/lbs
if [ $# -ne 3 ]; then
    read -p "weight in kg (lbs): " weight
    read -p "height in cm (in): " height
    read -n 1 -p "units kg/lbs: " units
    echo
else
    weight=$1
    height=$2
    units=${3:0:1}
fi
if [ $units = "k" ]; then
    ((bmi = $weight * 1000000 / ($height * $height)))
else
    ((bmi = $weight * 70300 / ($height * $height)))
fi
if [ $bmi -lt 1850 ]; then
    echo "underweight"
elif [ $bmi -lt 2500 ]; then
    echo "normal"
elif [ $bmi -lt 3000 ]; then
    echo "overweight"
else
    echo "obese"
fi
echo "scale=2;"$bmi/100" | bc -q
Browser

- Nowadays browsers are very powerful in their own right (can compute locally, not just communicate with a webserver)

CSS (Cascading Style Sheets)
SVG (Scalable Vector Graphics)
  - cf. HTML5 canvas
Javascript
  - *programming language*
DOM (Domain Object Model)
  - *programmatic access to documents*
Websockets
  - An API to interact with regular programs
HTML

• HTML: Hypertext Markup Language
  – Web browser: can read and render pages written in HTML
  – currently: HTML5
• What is "hypertext"?
  – linked content (Nelson, 1963)
  – nowadays: selected text/images/video can have arbitrary associated actions
• Hypercard for the Macintosh (1987)
• World Wide Web (WWW) (1992)
  – World Wide Web Consortium (W3C)
Reference

http://docs.webplatform.org/wiki/Main_Page
Client-side web development

• **HTML:**
  – structure of content
• **CSS (cascading style sheets):**
  – presentation
• **Javascript**
  – scripting language
• **DOM (document object model):**
  – hierarchical representation of webpage
• **SVG (scalable vector graphics):**
  – 2D graphical objects and methods
Client/server model

Web server scripting:
- cgi-bin: Common Gateway Interface
- PHP: Personal Home Page
- JSP: JavaServer Pages
- etc.

web browser:
- html
- Javascript
- SVG
- etc.

http://www.visualbuilder.com/jsp/tutorial/introduction-to-jsp/
### HTML

*boilerplate inserted by my emacs editor:*

```
1 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
2 <html> <head>
3   <title></title>
4 </head>
5 <body>
6   <h1></h1>
7 </body>
8 <hr>
9 <address></address>
10 <!-- hhmts start -->
11   <!-- hhmts end -->
12 </body> </html>
```

- **First line:**
  ```html
  <!DOCTYPE HTML>
  ```
  - signifies HTML5

- **Tags:**
  ```html
  <tag> ... </tag>
  ```
  - html
  - head: title, style, javascript definitions etc.
  - body: body of the document
  - h1: heading level 1 (1-6)
  - address: contact information

- **"self-closing" tags:**
  ```html
  <hr>
  <br/>
  ```
  - hr: horizontal rule
  - br: line break   optional: <br/>

- **optionally paired:**
  ```html
  <p> .. </p>: paragraph
  ```

- **comment:**
  ```html
  <!-- ... -->
  ```
HTML

- hypertext (link):
  - `<a href=URL>text</a>` (text presented in blue)

- URL: uniform resource locator
  - Examples:
    - `http://dingo.sbs.arizona.edu/~sandiway/`
    - `https://netid.arizona.edu/newid.php` (PHP)
    - `http://localhost/perl/test.pl` (mod_perl program)
  - Format:
    - `protocol://host(:port)/path`
    - `protocol://host(:port)/path?query`
      - protocol = `http` (hypertext transfer protocol)
      - port = TCP/IP port number
HTML

• Images:
  – `<img src=URL>`
  – attribute: `src` (required)
  – value: URL (or filename etc.) (jpg, gif, png supported, see note below)
  – attribute: `alt` (supposed to be required)
  – value: text
  – attribute: `height`
  – value: pixels
  – attribute: `width`
  – value: pixels
  – attribute: `align` (not in HTML5)
    – value: `top` | `bottom` | `middle` | `left` | `right`

• Can embed:
  – `<a href=URL><img src=URL></a>`

HTML

• Lists:
  – list item: `<li> ... </li>`
  – ordered lists: `<ol> ... </ol>`
  – unordered lists: `<ul> ... </ul>`
  – note: can be nested arbitrarily deep
HTML

• inline styling applied to text elements:
  – style="...; ...

    • font-size: Xpx
    • font-family: name, name ...
    • color: name (or hex RGB) e.g. #00CC00
    • background-color: name (or RGB)
    • text-align: left|right|center

  – note: serif, sans-serif, monospace are generic font families

  – note: X11 color names are okay,
HTML

• Other text element style tags:
  – `<em> ... </em>`                               *italics*
  – `<strong> .. </strong>`                        *bold*
  – `<pre> ... </pre>`                           *preformatted*
  – `<tt> ... </tt>`                            *monospaced*
HTML

• Tables:
  – `<table> ... </table>`
  – `<tr> ... </tr>` table row
  – `<th> ... </th>` table heading element
  – `<td> .. </td>` table data (one cell)
  – Attributes:
    • `border="size"` e.g. 1px
    • `colspan="number"` e.g. 2 (span next two columns)
    • `style="...;..."`
      – `border: width style color` also border-left, border-top, border-right, etc.
      – `border-width: top right bottom left`
      – `border-style: top right bottom left` e.g. type=solid|dotted|dashed|double|none
      – `border-color: top right bottom left`
      – `border-collapse: collapse|separate`
      – `padding: size`
      – `text-align: left | center | right`
      – `vertical-align: top | middle | bottom`
      – `width, height` e.g. 100px or 100%
  – Newer stuff:
    • `<thead> ... </thead>`
    • `<tbody> .. </tbody>`
HTML

• General chunks of html:
  – `<div style="..."> ... </div>` division
  – `<span style="..."> ... </span>` small chunks
Sample webpage

A webpage

First paragraph

Some text. Some more text. A lot more text is needed to see word-wrap. A link.

Google

A second paragraph.

1. First things first.
   ○ One
   ○ Two
2. Second things second.
3. Third things last.

Last modified: Thu Sep 18 17:28:57 MST 2014
Sample webpage

- Class:

```html
<html>
  <head>
    <title>A webpage</title>
  </head>
  <body>
    <h1>A webpage</h1>
    <table style="border: 1px solid blue">
      <tr>
        <td>1</td>
        <td>2</td>
        <td>3</td>
      </tr>
      <tr>
        <td>4</td>
        <td>5</td>
        <td>6</td>
      </tr>
      <tr>
        <td>7</td>
        <td>8</td>
        <td>9</td>
      </tr>
    </table>
    <h2>First paragraph</h2>
    <p>Some text. Some more text. <br> A lot more text is needed to see word-wrap. A <a href="http://google.com">link</a>.</p>
    <img width=100 src="https://www.google.com/images/srpr/logo11w.png"/>
    <p><br></p>
    <p style="font-family: bank gothic; font-size: 60; color: white; background-color: black">A second paragraph.</p>
    <ol>
      <li>First things first.</li>
      <ul>
        <li>One</li><li>Two</li>
      </ul>
    </ol>
    <li>Second things second.</li>
    <li>Third things last.</li>
  </body>
</html>
```