CSE 303, Winter 2006, Assignment 2
Due: Friday 27 January, 9:00AM

Last update: January 17 (late-day turnin)

You will debug a bash script and write a bash script. Problem 2 is more difficult than problem 1.

1. Get `mycat2buggy` from the course website. It is a buggy solution to the extra-credit from homework 1. Make `mycat2` by fixing `mycat2buggy`.

- Do *not* make large changes to the file; try to change as little as possible.
- There are 7–8 bugs (depending how you count) that can be fixed by changing, adding, or deleting a total of 16 characters.
- Recall `mycat2` cats its third-through-last arguments, sending stdout to its first argument and stderr to its second argument (exiting if there are fewer than three arguments or one of the first two arguments is an existing file). If the arguments have repeats (i.e., multiple arguments are the same string), it cats each file only once.

2. Write a shell script, described below, for creating a "private" web-page of photos downloaded from `www.kodakgallery.com` along with user-provided captions.

- **Motivation:** The KodakEasyShare Gallery[1] is a website that lets users share digital photos. When you "view an album" you get an HTML page that appears to require you to click on their fancy buttons to view the pictures. Your instructor prefers to have all the pictures on one simple HTML page and to enter captions for each picture. You will write a script to automate most of this conversion process.
- **Approximate Size:** The sample solution is 50 lines, including 16 lines that are blank or have only a comment, and another 17 lines that process command-line arguments. (This problem description is *much* longer than the solution.)
- **Specification:** Write a script called `kodakToMypage` that does the following.
  - Take two arguments. The first is a name of an HTML file *already saved* (manually) on the local computer (i.e., `attu`). The second is a name for a new subdirectory. If the script is not given exactly two arguments, or the first argument is a not a file, or the second argument already exists, exit with an appropriate error message.
  - *Assume* that the HTML file is a KodakEasyShare page, which (apparently) means:
    - For every picture in the corresponding album, there are two URLs, one for a big version of the picture and one for a small version.
    - The small-version URL starts `http://` and ends `SM.jpg`.
    - The big-version URL starts `http://`, ends `.jpg`, and does *not* end `SM.jpg`.
    - Each line of the file contains at most one URL.
  - Create a new directory (named the second argument) and put all new files in that directory.
  - Download the big version of each picture file. *Download no other files.*
  - Create a file `index.html` that is an HTML file (see separate documentation for a description of HTML and sample output) containing all the downloaded pictures and a caption for each.
  - Prompt the user for captions. In particular, for each picture do the following:
    - Display the picture (using the `display` program) and print `Enter Caption:` in the shell.
    - Use the `read` command to get a line of text from the user. You may *assume* the user enters only characters that have no special meaning in HTML (see the extra credit). Note: Use `read -e` to allow backspace to "work right".

[1] We have no connection to this web-site and are neither endorsing nor not-endorsing it. You need not become a user of the service to do this homework.

    * Stop displaying the picture.
    * Include the caption and (then) the corresponding *locally-saved* image in `index.html`.
    * You may use other "temporary" files (and probably will use at least one), but delete them before your script completes. (Note that while debugging it's helpful not to delete them.)

- **Advice and Hints:** Here is a detailed description of the sample solution, which uses some tricks. On attu, you can use `firefox` to view HTML. (Use Ctrl-o to open a local file.)
  - First process the command-line arguments as usual, using appropriate file-test operators.
  - Then create the new directory and `cd` into it.
  - Then use `grep` and `sed` to make a temporary file.
    * First get all the lines that have big-version URLs in them (perhaps by getting all lines with .jpg URLs and then getting only the subset that are *not* small-version URLs).
    * Then for all these lines, use `sed` to "match the whole line" but replace the part before the URL with "`wget `", the URL with itself, and the part after the URL with nothing.
    * Save the result in the temporary file.
  - Then download the photos by using `source` on the temporary file (man wget to see why this works). (Note: Redirect stdout and stderr to `/dev/null` or your script will look ugly to users. You may not want to do this while testing.)
  - Then use echo to produce the "beginning stuff" for `index.html`.
  - Then use a for-loop to process each picture (displaying it, getting a caption, and appending the appropriate HTML to `index.html`).
    * The files to loop through are exactly those ending in `.jpg`. On each iteration, do the following.
    * Run `display` as a background job.
    * Use `echo` and `read` (see 4.2 of the manual for `read`).
    * Then kill the background job (see `$!` in the manual).
    * Then write to `index.html`. You will need to write `<` and `>` characters; use double-quotes.
  - Then use echo to produce the "ending stuff" for `index.html`.
  - Finally remove any temporary files you made. (You should *not* delete the jpg files.)

- **Be Careful:** As programmers, you have the ability to write buggy programs that harm others. For example, if you write an infinite loop that constantly downloads files from `wwww.kodakgallery.com`, you (and possibly your instructor) could get in trouble. Please be careful!

3. **Extra Credit:** (Remember the course extra-credit policy. You may do either or both.)

- Change `kodakToMypage` so that the captions can include any character on the keyboard (e.g., `<` and `>`) and the HTML is still well-formed and includes the character.
- In a comment in `kodakToMypage`, explain why the instructor did not post the sample input on the course webpage and (for the same reason) why the HTML file you produce includes the *local* copies of the pictures. (Hint: This is not a technical question.)

Assessment: Your solutions should be

- Correct shell scripts that run on `attu.cs.washington.edu`
- In good style, including indentation and line breaks
- Of reasonable size

Turn-in Instructions: Use `turnin` for course cse303 and project hw2. In particular, type:

```
turnin -ccse303 -phw2 mycat2 kodakToMypage
```

from a directory containing your solution. If you use late-days, use project hw2late1 (for 1 late day) or hw2late2 (for 2) instead of hw2.