PHYS 410: Computational Physics Fall 2004 Homework 1
Due: Thursday, September 23, 10:00

Problem 1a: Create a ∼/.forward file on your lnx account so that mail sent to your userid at lnx[123] gets forwarded to your preferred e-mail address.

Problem 1b: Send me (choptuik@physics.ubc.ca) an e-mail message from your account on the Physics & Astronomy Department Sun server, physics.ubc.ca. Use the short form of the system's "hostname"—i.e. physics—for the subject of the message. Include *your* name in the body of the message. Incidentally, note that there is a Unix command hostname.

Problem 1c: Send me an e-mail message from your account on the lnx machines. Again, use the hostname of the specific machine (i.e. lnx1, lnx2 or lnx3) from which you send the message, as the subject, and include your name in the text of the message.

Problem 1d: Using the ssh-keygen command as discussed in class, and described in the *Unix* notes, create (or add to) the files ∼/.ssh/authorized_keys on *both* your physics and lnx accounts so that you can ssh and scp from physics to lnx1, lnx2, lnx3 and vice versa, without being prompted for a password.

Do the remaining problems using your account on the lnx *machines (it doesn't matter which one or ones you use). In particular, note that "∼" in the following refers to your* lnx *home directory, and all file and directory references are to files on those machines. As always, send me mail immediately if you encounter problems using any of the machines.*

IMPORTANT!! Be sure to read and follow all instructions *carefully*, as your grade will depend on certain files being in the right places with the right names. *Note that you will be copying files owned by user* phys410 *several times.*

Problem 2: I have created directories for each of you that you may use to "publish" Web pages (related to this course) via my research group's web server (http://laplace.physics.ubc.ca). Your personal directory (which I'll refer to as your Web directory) is ∼/public_html. I have created a "template" homepage named ∼phys410/public_html/index.html. Copy this HTML file into your Web directory (use the same name—index.html) and modify the HTML file to reflect your name, address, phone-number etc. Use your text editor of choice (presumably vi or emacs) to perform the modifications. If you don't feel like publishing any specific piece of information, specify it as "unlisted". Below the horizontal rule (line) in the template file, add suitably labelled links to (a) the course home page (b) the instructor's home page and (c) at least 5 Web pages that have something to do with "negative index of refraction metamaterials". Check your work by verifying that you can get to your home page by going to our Course page, selecting "Student Pages" and then your name. Also, please send me e-mail should the way I have listed your name in the "Student Pages" list need changing.

Problem 3a: I have created a subdirectory named `hw1` (for Homework 1) in your home directory. I have also changed the permissions on this directory so that your work remains private; please do not alter the permissions until after the homework has been graded. In this new directory, create another subdirectory named `a3`. In that directory (∼`/hw1/a3`), and using `vi` or `emacs`, create a file named `apple` that contains the following text from *Gravitation*, by Misner, Thorne and Wheeler. Try to duplicate the spacing, line breaks, punctuation etc. as closely as possible.

```
Once upon a time a student lay in a garden under an apple tree reflecting
on the difference between Einstein's and Newton's views about
gravity.  He was startled by the fall of an apple nearby.  As he
looked at the apple, he noticed ants beginning to run along its
surface. His curiousity aroused, he thought to investigate the
principles of navigation followed by an ant.  With his magnifying glass,
he noted one track carefully, and, taking his knife, made a cut in the apple
skin one mm above the track and another cut one mm below it.  He peeled
off the resulting little highway of skin and laid it out on the face
of his book.  The track ran as straight as a laser beam along this
highway.  No more economical path could the ant have found to cover
the ten cm from start to end of that strip of skin.  Any zigs and
zags or even any smooth bend in the path on its way along the apple
peel from starting point to end point would have increased its length.

   ``What a beautiful geodesic'', the student commented.
```

Problem 3b: Create a file in the same directory (∼`/hw1/a3`) called `kumquat` that is identical to `apple` except that all occurrences of the word "apple" are replaced with "kumquat". Leave a brief note in a file called `README` (again in the same directory) that describes how you created `kumquat` and made the changes.

Problem 4: As we will discuss in coming weeks, in the `Fortran 77` language, when one wants to continue a program statement across more than one line, one must use *continuation lines* following the first line of the statement. According to `Fortran 77`'s strict source-code formatting rules, a continuation line is signalled by five characters of white space at the beginning of a line, followed by an arbitrary non-white space character in the *sixth* column. For example, the following `Fortran 77` statement has a single continuation line:

```
      a = cos(x) *
     &    sin(x)
```

while this statement has two continuation lines

```
      a = cos(x) *
     1     sin(x) *
     2      tan(x)
```

In a directory ∼`/hw1/a4`, create a file called `continuation` which contains all of the continuation lines in the file ∼`phys410/hw1/q4/input.f`, in the order in which they appear in that file. You can assume (and verify) that the input file contains no `TAB` characters.

Leave a brief note in ∼`/hw1/a4/README` that describes how you solved the problem.

Problem 5: Make the directory ∼`/hw1/a5`. The file `/usr/share/dict/words` contains a list of "words" (mostly genuine English words—for the purposes of this question, any entry in the file will be deemed a "word"), one per line.

How many words does the file contain? Answer in ∼`/hw1/a5/README`.

In ∼`/hw1/a5` create files with the following names and contents (words should appear one per line):

1. `7letter` which contains all the seven-character (seven-letter) words in `/usr/share/dict/words` that do *not* begin with a vowel, in alphabetical order. (Define the set of vowels to be 'a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U', and 'y', 'Y')
2. `14letter` which contains an alphabetical list of all the fourteen-character words in `/usr/share/dict/words` that do *not* contain 'e', 'E', 'u' or 'U'.
3. `rab` which contains all the words in `/usr/share/dict/words` that begin with 'ab' in *reverse* alphabetical order (*Hint:* Use the Unix `sort` command).
4. `5cons` which contains an alphabetical list of all the words in `/usr/share/dict/words` which contain 5 or more *consecutive* consonants. (Define the set of consonants to be the complement of the vowel set used in part 1).

Leave comments in ∼`/hw1/a5/README` which describe how you solved each problem.

Problem 6: Make the directory ∼`/hw1/a6`. Use the plotting program `gnuplot` to produce a plot of $\cos(10x)\sin(8x)$ for $-2 \le x \le 1.5$. Save your plot as the postscript file `plot.ps` in the directory ∼`/hw1/a6`. *Hint:* Use `gnuplot`'s extensive on-line help: you may find `help plot`, `help postscript` and `help output` useful.

Problem 7: Make the directory ∼`/hw1/a7`. In that directory create a (Bourne)-shell script, `has`, which has the following usage:

```
usage: has [-i] pattern

   Prints names of text files rooted in current directory that contain 'pattern'

   Option:  -i -- ignore case in pattern matching
```

`has` has one required argument, which is a "grep-style" pattern (i.e. a regular expression): the script is to print the name of all *text* files rooted in the current directory (the directory in which the script is invoked) that contain the specified pattern one or more times. For the purpose of this exercise, a text file will *defined* as a file `f` such that the Unix command `file f` (see `man file`) generates output that matches the `grep` pattern `':.*text'` (note that the quotes are not part of this last pattern).

If the `-i` option is specified, then case is to be ignored in the pattern matching.

Sample usage:

```
% pwd
/d/lnx1/home/phys410/hw1/q7

% ls -lR
.:
total 16
drwxr-xr-x    2 phys410  phys410      4096 Sep  8 21:41 A/
-rw-r--r--    1 phys410  phys410        21 Sep  8 21:41 beatles
-rw-r--r--    1 phys410  phys410        22 Sep  8 21:41 doobie-bros
-rw-r--r--    1 phys410  phys410        23 Sep  8 21:41 prince
-rw-r--r--    1 phys410  phys410         0 Sep  8 21:45 typescript

./A:
total 8
-rw-r--r--    1 phys410  phys410        15 Sep  8 21:41 the-clash
-rw-r--r--    1 phys410  phys410        23 Sep  8 21:41 the-who

# Ensure that you understand what the following pipeline does, and how
# it does it.  See 'man find' and 'man pr' for information on the 'find'
# and 'pr' commands respectively

% find . -exec pr {} \; | grep -v '^[ ]*$'
pr: .: Is a directory
pr: ./A: Is a directory
2004-09-08 21:41                  ./A/the-clash                 Page 1
London Calling
2004-09-08 21:41                   ./A/the-who                  Page 1
Won't Get Fooled Again
2004-09-08 21:41                  ./doobie-bros                 Page 1
What a Fool Believes
2004-09-08 21:41                    ./beatles                   Page 1
The Fool on the Hill
2004-09-08 21:41                     ./prince                   Page 1
Nothing Can Compare 2U

% has
usage: has [-i] pattern

   Prints names of text files rooted in current directory that contain 'pattern'

   Option:  -i -- ignore case in pattern matching

% has Fool
./A/the-who
./doobie-bros
./beatles

# The following invocation generates NO output
% has fool
```

```
% has -i fool
./A/the-who
./doobie-bros
./beatles

% has '^The'
./beatles

% has '^The$'

% has -v
usage: has [-i] pattern

   Prints names of text files rooted in current directory that contain 'pattern'

   Option:  -i -- ignore case in pattern matching
```

As much as possible, your script should behave exactly as above; note in particular the usage message that is printed when the script is invoked with no arguments, or with invalid arguments. You can and should require that the `-i` option, if present, precede the required `pattern` argument.

Hints: Consider the *Unix* commands: `find`, `file`, and `grep` and note that you may find it convenient to write an additional "helper" script which is invoked by `has`.

Recall that you can use the `-x` flag in script headers for debugging purposes:

```
#! /bin/sh -x
```

Note that the above example does not involve any files that are *not* text according to the *file* command. You should ensure that your script does *not* output the names of non-text files. In testing this feature of `has`, the non-text file

```
   /d/lnx1/home/phys410/hw1/q7a
```

that contains the string `sort`, may be of use to you.

```
% pwd
/d/lnx1/home/phys410/hw1/q7a

% ls
grep-sort-matches*

% file grep-sort-matches
grep-sort-matches: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
  for GNU/Linux 2.2.5, dynamically linked (uses shared libs), stripped

% grep sort grep-sort-matches
Binary file grep-sort-matches matches
```

Once you have coded and debugged (if necessary!) `has`, create the directory `~/bin`, *write protect* that directory using the command `chmod go-rwx ~/bin`, and copy `has`—well as any helper scripts you have coded for your solution—there. Once this is done, execute the C-shell built-in command `rehash` to update the command hash tables, and ensure that `has` works as a regular command for you; for example, verify that you get a usage message from `has` when you invoke it from your home directory:

```
% cd; has
usage: has [-i] pattern

   Prints names of text files rooted in current directory that contain 'pattern'

   Option:  -i -- ignore case in pattern matching
```