PHYS 410: Computational Physics Fall 2005 Homework 1
Due: Thursday, September 22, 10:00 AM

Last Updated: Sunday, September 11, 11:00 AM

PLEASE e-mail bug reports, comments, gripes etc. to Matt: `choptuik@phas.ubc.ca`

NOTE: The notation/acronym "a.k.a", means "also known as", i.e. "is an alias for".

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: Create and/or modify your `/mail/$HOME/.procmailrc` file on your `physics` account so that mail sent to your userid at `physics` gets forwarded to your preferred e-mail address.

Problem 1c: 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`.

IMPORTANT NOTE: The command `Mail` covered in class appears to be broken on `physics.ubc.ca` at the current time. Thus, please do *do not* use `Mail` on `physics` to complete this part of the assignment. Rather, I suggest that you use `pine`. On the `lnx` machines, you can use *either* `Mail` or `pine` (or any other mail client that you can find, for that matter) as you wish. I apologize for the inconvenience caused to any student who attempted to use `Mail` on `physics`, which no doubt resulted in an obscure error message, and non-delivery of the mail.

Problem 1d: 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 1e: 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 the following, let us define *alphabetical order* as the default order used by the Unix `sort` command.

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

1. `9let1vow` that lists, in alphabetical order, all of the 9-letter words containing precisely one (1) vowel. Define the set of vowels to be [aAeEiIoOuUyY].
2. `8let2vow` that lists, in *reverse alphabetical order*, all of the 8-letter words containing precisely two (2) vowels. Define the set of vowels as above. *Hint consider the Unix* `sort` *command.*
3. `5cons-ng` that lists, in alphabetical order, all words that contain at least one string of 5 or more consecutive consonants, *none of which can be an "n" or a "g"*. Note that this does *not* necessarily mean that there cannot be a "n" or "g" in the word!

Leave comments in ~`/hw1/a5/README` **that describe how you solved each of the above three sub-problems.**

Finally, using (1) your answers to the above sub-problems, (2) Unix's backquote mechanism (which lets us represent the output of a command using the invocation command per se), and (3) the quirky but venerable `expr` command, write a "one-liner" that computes how many *more* words there are in case 2 above than in case 1. Note that by a "one-liner", I mean a *single* Unix command (or pipeline of commands). Furthermore, the "one-liner" *cannot* use any shell variables, and it *cannot* refer to *any* file other than `/usr/share/dict/words`. Using what should now be your beloved editor (i.e. either `vi` or `emacs`), record your one-liner in the file ~`/hw1/a5/thesource`. Then, assuming that your working directory is ~`/hw1/a5`, the command

```
lnx% source thesource
```

should display, on standard output, the (integer-valued) answer to the question regarding the difference in cardinality of cases 2 and 1.

Problem 6: Make the directory ~`/hw1/a6`. Use the plotting program `gnuplot` to produce a plot of $\cos(3x)\sin(4x)\cos(5x)$ for $-2 \leq x \leq 3.5$. 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, `get-nasa-iod`, which has the following usage:

```
usage: get-nasa-iod [-q] image-number

   Retrieves one of NASA's Images of the Day using 'wget'.

   image-number -- Integer between 1 and 401
   Option:   -q -- Suppress output from 'wget'
```

`get-nasa-iod` uses the very powerful data retrieval command, `wget`—commonly found on up-to-date Unix systems—to fetch *directly* one of NASA's "Image(s) of the Day"; see

```
   http://www.nasa.gov/multimedia/imagegallery/
```

for more information on the "Image of the Day" gallery, and note that in the following I will make frequent use of the acronym IOD for "Image of the Day".

Let us first be clear about the purpose of the script.

NASA has all of these great images that they've hand selected from the half a bizillion or so merely wonderful images they have in stock, but they package and export these images—which are stored as JPEG (Joint Photographic Experts Group) files—via *links* to the images, embedded in what amounts to "cover pages" for the pictures. Now, there are about 400 "Images of the Day", and, when done, your script will take a *single integer*, and go straight to the heart of the matter, retrieving *only* the way cool image that corresponds to that specific IOD—i.e. just the `.jpg` file, ma'am—with none of the artery-clogging and thus potentially unhealthy HTML that's nominally wrapped around it—i.e. what you get when the image is served up the way NASA wants to serve it up to you!

Before proceeding to a "definition by example" for `get-nasa-iod`, it is instructive to see `wget` in action in the context of a simpler example.

As you are all aware (and if you aren't, you are now!), the document you are reading "lives online" in the very fundamental sense that, for example, any printed copy that I distribute in class will have been produced *using a web browser* that was pointing to some specific HTML page, or some other file (PS, PDF, ...) that I'm maintaining for this course. In Web parlance, we say that this document has *a URL* (one of those exceptions to using "an" in front of something starting with a vowel, with URL $\equiv$ Uniform Resource Locator). Specifically, the URL for this handout is

```
http://laplace.physics.ubc.ca/410/HW/hw1_0.ps
```

and thus you can download a copy of what you are reading, or rather, a copy of what you are reading *modulo any changes that I've made to the "master" document in the time since what you are reading was printed.*

So go ahead! The instructor dares you!!

If it is convenient, log into a Linux system, or some other system that has `wget` installed, and download *this very document* using the command $\cdots$

```
wget http://laplace.physics.ubc.ca/410/HW/hw1_0.ps
```

$\cdots$ drum roll, please!! $\cdots$ and *be sure that you pay careful attention to* case *in your typing of the URL, since URLs* are *case-sensitive!*

If that worked, and you haven't used `wget` before, then ideally you should have experienced a small warm glow, or perhaps even slight tingling, as the download proceeded "automatically". If not, well, that's about as exciting as this part of the course gets, I'm sorry to say!

As you might imagine, `wget` has many options and can handle pretty well all of your network slicing, dicing and juicing needs, but single argument (single URL) invocations such as the above are all that are required to complete this exercise.

Sample invocations and resulting output:

As the old saying goes, "an invocation is worth a thousand words", i.e., it is best to describe/define the functionality of `get-nasa-iod` through sample invocations, along with (1) the output from the script, and (2) the image files that are downloaded from NASA, both of which may result from those invocations.

Your mission is to make your script behave just like the *key script* (i.e. the instructor's implementation).

1. *Invocation with no (0) arguments. Usage message is triggered.*

```
lnx1% get-nasa-iod

usage: get-nasa-iod [-q] image-number

   Retrieves one of NASA's Images of the Day using 'wget'.

   image-number -- Integer between 1 and 401
   Option:   -q -- Suppress output from 'wget'
```

2. *Invocation with one (1) valid argument. Script first retrieves URL*

   `http://www.nasa.gov ... /image_feature_011.html`

   then retrieves URL

   `http://www.nasa.gov/images/content/1428main_MM_Image_Feature_11_rs3.jpg`

```
lnx1% get-nasa-iod 11

--08:55:45--  http://www.nasa.gov/multimedia/imagegallery/image_feature_011.html
           => `.get-nasa-iod'
Resolving www.nasa.gov... 65.110.21.226, 65.203.232.4
Connecting to www.nasa.gov[65.110.21.226]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19,862 [text/html]

    0K .......... .........                                  100%  497K

08:55:45 (494.72 KB/s) - `.get-nasa-iod' saved [19862/19862]

--08:55:46--  http://www.nasa.gov/images/content/1428main_MM_Image_Feature_11_rs3.jpg
           => `NASA-IOD-011.jpg'
Resolving www.nasa.gov... 65.110.21.226, 65.203.232.4
Connecting to www.nasa.gov[65.110.21.226]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 62,090 [image/jpeg]

    0K .......... .......... .......... .......... .......... 82%  633K
   50K ..........                                            100%  967K

08:55:46 (670.95 KB/s) - `NASA-IOD-011.jpg' saved [62090/62090]

get-nasa-iod: Input argument '11' was encoded as 011
get-nasa-iod:

-rw-r--r--  1 matt choptuik 62090 Jun 13  2003 NASA-IOD-011.jpg
```

3. *Invocation with another single valid argument. Note that this time the argument (i.e. the number 86) is encoded in the* `image_feature....html` *file as* `'86'`*, rather than* `'086'`*. Pretty tricky, NASA web-meisters!*

```
lnx1% get-nasa-iod 86

--08:55:46--  http://www.nasa.gov/multimedia/imagegallery/image_feature_086.html
           => `.get-nasa-iod'
```

```
Resolving www.nasa.gov... 65.110.21.226, 65.203.232.4
Connecting to www.nasa.gov[65.110.21.226]:80... connected.
HTTP request sent, awaiting response... 404 Not Found
08:55:51 ERROR 404: Not Found.

--08:55:51--  http://www.nasa.gov/multimedia/imagegallery/image_feature_86.html
           => `.get-nasa-iod'
Resolving www.nasa.gov... 65.203.232.4, 65.110.21.226
Connecting to www.nasa.gov[65.203.232.4]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20,421 [text/html]

    OK .......... .........                                  100%  464K

08:55:51 (455.05 KB/s) - `.get-nasa-iod' saved [20421/20421]

--08:55:51--  http://www.nasa.gov/images/content/49705main_MM_Image_Feature_86_rs4.jpg
           => `NASA-IOD-086.jpg'
Resolving www.nasa.gov... 65.203.232.4, 65.110.21.226
Connecting to www.nasa.gov[65.203.232.4]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 43,695 [image/jpeg]

    OK .......... .......... .......... .......... ..        100%  628K

08:55:52 (622.13 KB/s) - `NASA-IOD-086.jpg' saved [43695/43695]

get-nasa-iod: Input argument '86' was encoded as 86
get-nasa-iod:

-rw-r--r--  1 matt choptuik 43695 Sep  2  2003 NASA-IOD-086.jpg
```

4. *Invocation with two (2) valid arguments. As for ALL valid two argument calls, the first argument in this case is '-q', which results in the suppression of all output from* `wget`*. The script thus does its downloading from the NASA site* silently*, and generates only (1) the diagnostic message telling us how many digits were used in the encoding of the argument integer, and (2) an* `ls` *listing of the (renamed)* `.jpg` *file; both are written to standard output.*

```
lnx1% get-nasa-iod -q 99

get-nasa-iod: Input argument '99' was encoded as 99
get-nasa-iod:
-rw-r--r--  1 matt choptuik 44279 Nov 26  2003 NASA-IOD-099.jpg
```

5. *Invocation with a single* invalid *argument, 0, which is not in the valid range 1 to 401 inclusive. Usage message is triggered.*

```
lnx1% get-nasa-iod 0

usage: get-nasa-iod [-q] image-number

   Retrieves one of NASA's Images of the Day using 'wget'.

   image-number -- Integer between 1 and 401
   Option:   -q -- Suppress output from 'wget'
```

6. *Invocation with a single* invalid *argument, 402, which is not in the valid range 1 to 401 inclusive. Usage message is triggered.*

```
lnx1% get-nasa-iod 402

usage: get-nasa-iod [-q] image-number

   Retrieves one of NASA's Images of the Day using 'wget'.
```

```
  image-number -- Integer between 1 and 401
  Option:   -q -- Suppress output from 'wget'
```

7. *Invocation with what is ostensibly a valid single argument. Nonetheless, in this instance the script fails to download the 28th IOD, since, at least as closely as the instructor has looked, 28 is one of 10* exceptional *(as I will call them) integers in the range 1 to 401 inclusive, for which there is* no IOD. *The last part of this part of the homework, and thus of the entire homework per se has you determine the other 9 exceptional numbers.*

```
lnx1% get-nasa-iod 28

--08:55:53--  http://www.nasa.gov/multimedia/imagegallery/image_feature_028.html
           => `.get-nasa-iod'
Resolving www.nasa.gov... 65.110.21.226, 65.203.232.4
Connecting to www.nasa.gov[65.110.21.226]:80... connected.
HTTP request sent, awaiting response... 404 Not Found
08:55:53 ERROR 404: Not Found.

--08:55:53--  http://www.nasa.gov/multimedia/imagegallery/image_feature_28.html
           => `.get-nasa-iod'
Resolving www.nasa.gov... 65.203.232.4, 65.110.21.226
Connecting to www.nasa.gov[65.203.232.4]:80... connected.
HTTP request sent, awaiting response... 404 Not Found
08:55:58 ERROR 404: Not Found.

get-nasa-iod: Could not download 'http://www.nasa.gov/multimedia/imagegallery/image_feature_28.html'
```

8. *The following is NOT an invocation, but rather a "long" (*`-l`*) listing of the* `.jpg` *files that the sample invocations have downloaded thus far. Note that by using a relabelling scheme with a width-three (3) field with leading zeros displayed, our files are automatically listed in* numerical *order and* lexicographical *(or* collating sequence *order).*

   Which order does `ls` *use—numerical or lexicographical?* **ANSWER in** ~`/hw1/a7/README`.

```
lnx1% /bin/ls -l *.jpg

-rw-r--r--  1 matt choptuik 62090 Jun 13  2003 NASA-IOD-011.jpg
-rw-r--r--  1 matt choptuik 43695 Sep  2  2003 NASA-IOD-086.jpg
-rw-r--r--  1 matt choptuik 44279 Nov 26  2003 NASA-IOD-099.jpg
```

Again, as much as possible, your script should behave *exactly* **as above, when invoked as above.** Note in particular (1) the usage message that is printed when the script is invoked with no arguments, or with invalid arguments, (2) the various diagnostic messages—typically prefaced with `get-nasa-iod:`—that appear with successful and unsuccessful downloads alike, and (3) the fact that you can and should require that the `-q` option, if present, precede the required integer-valued argument, `image_number`.

Hints, Notes and Additional Questions to be Answered

1. Study the above sample output, and visit the NASA IOD site to glean as much information as you need concerning the URLs that you must download. This is how the instructor got going on his solution to the problem.

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

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

3. Most Unix systems, including Linux and Sun-OS (Solaris) implement the command `printf` (see `man printf`) which has the same name, and essentially the same function and syntax as its `C`-language counterpart, `printf` (see `man 3 printf`). This command plays a leading role in the instructor's

implementation of `get-nasa-iod`, and is highly commended to you. Note that `man`—and you!—distinguish between commands/functions/... with the same name (i.e. an "overloaded" name) via the manual *section*, i.e. the optional, integer valued first argument to `man` (see `man man`!).

How does one determine in how many sections in the manual there is a description for a command/function/... etc. for `printf`? **ANSWER in the `README` file that you have previously created for this question.**

Why do I go on about the different versions of `printf` and their corresponding `man` sections?

Because you quite probably *will* find "Sec. 1 `printf`" (the command) useful in completing this question but, on Linux systems, will get *no* information about the *decidedly non-trivial syntax* of "Sec. 1 `printf`" in the Sec. 1 `man` page for `printf` per se. Rather, presumably since "Sec. 1 `printf`" is designed to emulate, in both syntax and functionality, "Sec. 3 `printf`", whomever is/was responsible for the Sec. 1 `man` page was arguably a little lazy, and one must refer to the `man` page for "Sec. 3 `printf`" to learn the syntax to be used with its Sec. 1 cousin. Those of you who know `C` should have a bit of a headstart here; the rest of you may find yourselves having to study *both* `man` pages a bit. As always though, experimentation with a few examples is also a powerful approach—remember, almost all respectable Unix commands are designed to be used *interactively!*

4. Observe that many sites, particularly popular ones such as NASA, have mechanisms in place that serve as a first line of defense against rapid-fire downloads, such as the type that could result from indiscriminate use of the script that you are to write.
   Thus, you will find that the `.jpg` files associated with the various IODs are *not* stored in the NASA site in some "logical" (i.e. easily guessable) fashion such as

```
NASA-Image-of-the-day-001.jpg
NASA-Image-of-the-day-002.jpg
             .
             .
             .
NASA-Image-of-the-day-400.jpg
NASA-Image-of-the-day-401.jpg
```

   Rather, if I were to ask for the images associated with the 1st, 157th, and 302nd IODs, I would be served up the files:

```
1424main_MM_Image_Feature_10_mm3.jpg
57640main_image_feature_157_jw4.jpg
112414main_image_feature_302_ajh_4.jpg
```

   respectively.

   Thus, although the `image_number` (i.e. which of the 401 or so IODs is this?), namely 1, 157 and 302 for the above files *is*, loosely speaking, encoded in the file name, there is apparently *also a 4-, 5-, or 6-digit random number prepended to the image name.* This provides the aforementioned first line of defense against naive script-writers ("script kiddies" and other species of not-native-to-UBC mammals).

   Vis a vis the IODs, however, NASA has *only* that line of defense, plus a slight wrinkle alluded to in the 7th usage example above. This is in contrast to some other busy sites, such as the physics electronic e-print archive, `http://www.arxiv.org`, which prohibits *any* retrievals via `wget`; otherwise you surely would be coding a script for this homework that downloaded from *that* site, since the information therein is *much* more interesting in total than the NASA IOD collection, if not so pretty to look at!

   We can easily work around NASA's defense by nothing that a link to each distinct IOD (i.e. the URL for the actual `.jpg` image for that IOD) can be found in what is essentially an HTML "cover page" for that image, and that, happily, these cover pages have URLs that *do* follow a (mostly) logical naming sequence.

Namely, the cover pages of the IODs that the instructor has been able to track down are either of the form

```
http://www.nasa.gov/multimedia/imagegallery/image_feature_001.html
http://www.nasa.gov/multimedia/imagegallery/image_feature_002.html
http://www.nasa.gov/multimedia/imagegallery/image_feature_003.html
                               .
                               .
                               .
http://www.nasa.gov/multimedia/imagegallery/image_feature_401.html
```

or

```
                               .
                               .
http://www.nasa.gov/multimedia/imagegallery/image_feature_86.html
                               .
                               .
http://www.nasa.gov/multimedia/imagegallery/image_feature_99.html
                               .
                               .
                               .
```

Thus your scripts will likely invoke `wget` *at least twice* for every valid invocation of the script per se, at least once to retrieve the appropriate one of the files

```
http://www.nasa.gov/multimedia/imagegallery/image_feature_999.html
http://www.nasa.gov/multimedia/imagegallery/image_feature_99.html
```

where the `9`'s in the above denote *generic* integers, and a second or third time to retrieve the actual JPEG image, (a.k.a. the actual `.jpg` file) the URL for which is contained in the `.html` file. (Note that `.html` files are plain-text (ASCII/Unicode) files.)

5. There's another twist to this assignment in that, for whatever reason, there are 10 integers between 1 and 401 for which no corresponding `image_feature_···.html` files seem to exist. 28 is such an integer, as the 7th invocation above illustrates.

   In addition to coding this script, you must determine the other 9 integers between 1 and 401 which are "exceptional" in this regard. **ANSWER in the `README` file for this question.**

6. As well as the `wget` and `printf` commands, you may find `grep` and/or `sed` useful in the coding of your script. Review and/or study the course Unix notes for these commands and/or use the `man` pages and/or use the Web as necessary.

Once you have coded and debugged (if necessary!) `get-nasa-iod`, create the directory `~/bin`, *write protect* that directory using the command `chmod go-rwx ~/bin`, and copy `get-nasa-iod`—well as any helper scripts you may 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 `get-nasa-iod` works as a regular command for you; for example, verify that you get a usage message from `get-nasa-iod` when you invoke it from your home directory:

```
% cd; get-nasa-iod

usage: get-nasa-iod [-q] image-number

   Retrieves one of NASA's Images of the Day using 'wget'.

   image-number -- Integer between 1 and 401
   Option:   -q -- Suppress output from 'wget'
```

The grader (and possibly the instructor) will test and evaluate your script using our own series of invocations.