CSC 209 Assignment 1, Fall 2013: Shell scripts

Due by the end of Friday October 11, 2013; no late assignments without written explanation.

Please re-read the statement about academic offences on the course information sheet!

1. A computational shell script

Write a shell script named *average* which computes the mean of the input numbers, which can be in files on the command-line, or on the standard input, in the manner of a standard unix "software tool".

The numbers can be separated by any white space, so you might want to use 'tr' to canonicalize the data. The input numbers must be integers (but might be negative).

Example uses, where '%' is the shell prompt:

```
% echo 5 | sh average
5
% echo 5 7 | sh average
6
% echo -5 -7 | sh average
-6
% sh average /u/ajr/t2
3047
% sh average file1 file2 file3
33
% sh average /dev/null
average: no input values
%
```

Additionally, you might have an error message indicating non-numeric input. (It might be some other tool, such as expr, which gives the non-numeric input error message.)
It is acceptable to use integer division; e.g. the average of 5 and 8 would be 6.
It is acceptable to fail to notice the non-numeric input instead of diagnosing it.

2. A data system implemented by shell scripts

It is common when connecting one's computer to a new network to have to "register" the computer with system administration staff. Typically this means you automatically get an IP address via the DHCP protocol when you connect your computer to the network, and in some cases this registration also gives you a name in the Domain Name System.

If your computer has hostname "foo", this name would be supported by a line in a DNS "zone" file which looks like this:

```
foo   IN   A   192.168.1.38
```

and if that computer automatically gets that IP address by DHCP then that would be supported by a clause in the dhcp server configuration file which looks like this:

```
host foo {
    hardware ethernet 00:0d:93:3c:17:f0;
    fixed-address 192.168.1.38;
}
```

Your task for the rest of this assignment is to write a shell script named "addhost" which adds a new computer to the database given its hardware address, choosing an IP address for it; a shell script named "makedns" which outputs a DNS zone file based on the hosts (computers) registered with addhost; and a shell script named "makedhcp" which outputs a DHCP server configuration file similarly.

addhost

The "addhost" script expects two command-line arguments: the machine name and the hardware address, normally known as the "MAC address" (nothing to do with Apple Macintosh). It will store its information in the directory $HOME/hosts, which should exist (it is a fatal error if it doesn't). Each file in $HOME/hosts has a file name which is the machine name (e.g. "foo" in the above example) and two lines of content. The first line is the MAC address, which is a colon-separated list of six eight-bit numbers in hexadecimal (for the purposes of this assignment, just accept any string), and the second line is the assigned IP address.

(continued)

Assigned IP addresses will start with 192.168.1.1. To assign a new IP address, examine all of the files in $HOME/hosts (if any), find the maximum third octet (number), and find the maximum fourth octet for that third octet. Add one to the fourth octet, but 255 (and greater) is not allowed; when the fourth octet reaches 255, reset it to 1 and increment the third octet.
Examples: The IP address subsequent to 192.168.3.123 is 192.168.3.124; the IP address subsequent to 192.168.3.254 is 192.168.4.1.

Note that you must consult the $HOME/hosts/* files to determine the next IP address, rather than storing the current value elsewhere, because there may be programs other than your addhost which modify the $HOME/hosts database.

makedns

A DNS zone file looks something like this (the following is slightly simplified for this assignment):

```
example.org IN SOA ns1.example.org 2013091502
    IN NS 0 ns1.example.net.
    IN NS 0 ns2.example.net.
    IN MX 0 mail.example.net.
foo IN A 192.168.1.1
bar IN A 192.168.1.2
baz IN A 192.168.1.3
```

The most exciting part here is the "serial number" at the end of the first line. Each new version of the zone file needs to have a higher serial number than the previous one, so you will store the last serial number in the file $HOME/lastserial, and consult this when generating the new file.

Furthermore, we format the serial number as yyyymmddnn for the current date followed by two more digits as described shortly. Those first eight digits can be generated via "date +%Y%m%d". The subsequent two digits will initially be zeroes, but if "makedns" is run multiple times in the same day, larger numbers will be used (just add one). If makedns is run more than a hundred times in the same day, we will end up carrying into the day number, which is fine; just keep track of the last serial number.

The rest of the prologue (the first four lines above) you can just output verbatim.

Then you output the host records, one per line.

Makedns outputs its data to the standard output, for the user to redirect as appropriate (or for you simply to view while testing).

makedhcp

The DHCP server configuration file is relatively simpler. It looks like this:

```
authoritative;
deny unknown-clients;
option routers 10.3.8.16;
option domain-name-servers 10.3.8.20, 10.3.8.21;

subnet 192.168.0.0 netmask 255.255.0.0 {
}

host foo {
    hardware ethernet 00:0d:93:3c:17:f0;
    fixed-address 192.168.1.1;
}
host bar {
    hardware ethernet 00:0d:93:3c:17:f8;
    fixed-address 192.168.1.2;
}
host baz {
    hardware ethernet 00:0d:93:3c:17:fc;
    fixed-address 192.168.1.3;
}
```

(continued)

The "makedhcp" program outputs this data to the standard output. The first two paragraphs are fixed text, *not* simplified for this assignment.

To submit

I suggest you begin by making a subdirectory to hold your assignment files, plus any other associated files and working copies. You should call your assignment files average, addhost, makedns, and makedhcp.

Once you are satisfied with your files, you can submit them for grading with the command

```
submit -c csc209h -a a1 average addhost makedns makedhcp
```

You may still change your files and resubmit them any time up to the due time. You can check that your assignment has been submitted with the command

```
submit -l -c csc209h -a a1
```

You can submit files individually instead of all at once, and you can resubmit a particular file by using the –f option, e.g.

```
submit -c csc209h -a a1 -f addhost
```

This is the only submission method; you do not turn in any paper.

Other notes

For now, we will run our shell scripts by typing "sh file" or "sh file args ...".

Shell scripts should begin with a minimal assignment to the PATH variable as discussed in class. But we will not be worrying about the "#!/bin/sh" thing for assignment one.

Do not use awk, perl, python, or any other programming language in your scripts besides *sh*. (These are valuable languages to know, but are not the point of the current assignment, in which you will be graded on your *sh* programming.)

Please see the assignment Q&A web page at

```
http://www.cdf.toronto.edu/~ajr/209/a1/qna.html
```

for this and other reminders, and answers to common questions.

Remember:

This assignment is due at the end of Friday, October 11, by midnight. Late assignments are not ordinarily accepted and *always* require a written explanation. If you are not finished your assignment by the submission deadline, you should just submit what you have, for partial marks.

Despite the above, I'd like to be clear that if there *is* a legitimate reason for lateness, please do submit your assignment late and send me that written explanation.

I'd also like to point out that even a zero out of 10% is far better than cheating and suffering an academic penalty. Don't cheat even if you're under pressure. Whatever the penalty eventually applied for cheating, it will be worse than merely a zero on the assignment.