

# Program 1 – CS 344

This assignment asks you to write a bash shell script to compute statistics. The purpose is to get you familiar with the Unix shell, shell programming, Unix utilities, standard input, output, and error, pipelines, process ids, exit values, and signals (at a basic level). What you're going to submit is your script, called simply "stats".

## Overview

In this assignment, you will write a bash shell script to calculate mean averages and medians of numbers that can be input to your script from either a file or via stdin. This is the sort of calculation I might do when figuring out the grades for this course. The input will have whole number values separated by tabs, and each line will have the same number of values (for example, each row might be the scores of a student on assignments). Your script should be able to calculate the mean and median across the rows (like I might do to calculate an individual student's course grade) or down the columns (like I might do to find the average score on an assignment).

You will probably need commands like these, so please read up on them: `while`, `cat`, `read`, `expr`, `cut`, `head`, `tail`, `wc`, and `sort`.

Your script must be called simply "stats". The general format of the stats command is:

```
stats {-rows|-cols} [input_file]
```

Note that when things are in curly braces separated by a vertical bar, it means you should choose one of the things; here for example, you must choose either `-rows` or `-cols`. The option `-rows` calculates the mean and median across the rows; the option `-cols` calculates the mean and median down the columns. When things are in square braces it means they are optional; you can include them or not, as you choose. If you specify `input_file` the data is read from that file; otherwise, it is read from stdin.

Here is a sample run of what your script might return, using an

input file called "test\_file" (this particular one can be downloaded [here](#), note that in Windows, the newline characters may not display as newlines. Move this to your UNIX account, without opening and saving it in Windows, and then cat it out: you'll see the newlines there). While the representation of the data file below uses HTML spaces to *look* like tab characters, the actual data used in this assignment uses tabs.

```
$ cat test_file
1      1      1      1      1      1      1
39     43     4      3225  5      2      2
6      57     8      9      7      3      4
3      36     8      9      14     4      3
3      4      2      1      4      5      5
6      4      4814  7      7      6      6
$ stats -rows test_file
Average Median
1      1
474    5
13     7
11     8
3      4
693    6
$ cat test_file | stats -c
Averages:
10     24     806    542    6      4      4
Medians:
6      36     8      9      7      4      4
$ echo $?
0
$ stats
./stats {-rows|-cols} [file]
$ stats -r test_file junkwucnqwiuecnqiwe
./stats {-rows|-cols} [file]
$ stats -both test_file
./stats {-rows|-cols} [file]
$ chmod -r test_file
```

```
$ stats -columns test_file
./stats: cannot read test_file
$ stats -c no_such_file
./stats: cannot read no_such_file
$ echo $?
1
```

## Specifications

You must check for the right number and format of arguments to stats. You should allow users to abbreviate -rows and -cols; any word beginning with a hyphen and then a lowercase r is taken to be rows and any word beginning with a hyphen and then a lowercase c is taken to be cols. So, for example, you would get mean averages and medians across the rows with -r, -rowwise and -rumplestiltskin, but not -Rows. If the command has too many or too few arguments or if the arguments are of the wrong format you should output an error message to standard error. You should also output an error message to stderr if an input file is specified, but is not readable.

You must output the statistics to stdout in the format shown above. Be sure all error messages are sent to stderr, including the "usage" returned above when someone doesn't specify the correct parameters to stats. If a specified input file is empty, this is not an error: do not output any numbers or statistics. In this event, either send an informational message to stderr and exit, or just exit. If there are any errors of any kind (remember an empty input file is not an error), then the exit status should be set to 1; if the stats program runs successfully, then the exit value should be 0. Your stats program should be able to handle data with any reasonable number of rows or columns; however you can assume that each row will be less than 1000 bytes long (because Unix utilities assume that input lines will not be too long), but don't make any assumptions about the number of rows. Think about where in your program the size of the input matters. You can

assume that all rows will have the same number of values; you do not have to do any error checking on this.

Though optional, I do recommend that you use temporary files; arrays are not recommended. For this assignment, any temporary files you use should be put in the current working directory. (A more standard place for temporary files is in /tmp but don't do that for this assignment; it makes grading easier if they are in the current directory.) *Be sure any temporary file you create uses the process id as part of its name, so that there will not be conflicts if the stats program is running more than once.* Be sure you remove any temporary files when your stats program is done. You should also use the trap command to catch interrupt, hangup, and terminate signals to remove the temporary files if the stats program is terminated unexpectedly.

All values and results are and must be whole numbers. You may use the `expr` command to do your calculations, or any other bash shell scripting method. Do not use any other languages other than bash shell scripting: this means that, among others, awk, sed, tcl, bc, perl, & the python languages and tools are off-limits for this assignment. Note that `expr` only works with whole numbers. When you calculate the average you must round to the nearest whole number, where half values round up (i.e. 7.5 rounds up to 8). This is the most common form of rounding. When doing truncating integer division, this formula works to divide two numbers and end up with the proper rounding:

$$(a + (b/2)) / b$$

You can learn more about rounding methods here (see Half Round Up):

<http://www.mathsisfun.com/numbers/rounding-methods.html>  
(Links to an external site.)

[Links to an external site.](#)

To calculate the median, sort the values and take the middle value. For example, the median of 97, 90, and 83 is 90. The

median of 97, 90, 83, and 54 is still 90 - when there are an even number of values, choose the *larger* of the two middle values.