

CSc 332 (L) - Operating Systems

Lab - Spring 2018

Task 4 - Average Grade Calculator

Assigned March 2 2018

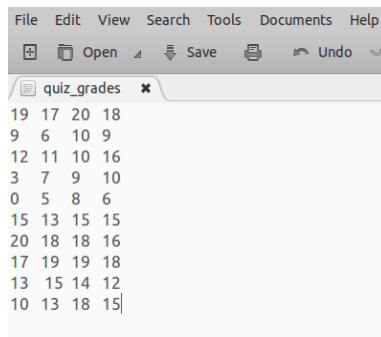
Max Points: 30 Due: March 16, 2018 11:59PM

Average Grade Calculator

There are 10 students enrolled in a course. The course covers x number of chapters from a textbook ($x > 1$). In each chapter y number of homework(s) are assigned ($y \geq 1$). The average grade for each homework in all the chapters need to be found out.

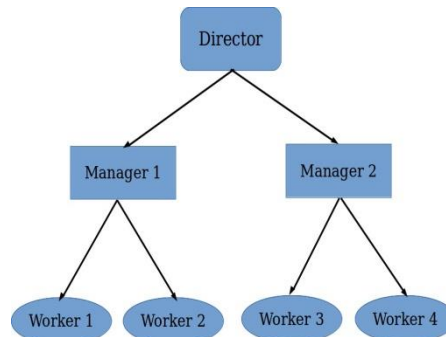
To solve this, write program which has the main process as *Director* process, which reads a file containing grades of all homework assignments of all chapters and creates x number of *Manager* processes. Each *Manager* process will take care of solving a chapter. Each manager process will create y number of *Worker* process and pass one homework to each of them and they calculate and print the average.

The input file should contain the data according to the value of x and y . For example, the input text file and the process tree for $x = 2$ and $y = 2$ will look like the following:



Row	Col 1	Col 2	Col 3	Col 4	Col 5
1	19	17	20	18	
2	9	6	10	9	
3	12	11	10	16	
4	3	7	9	10	
5	0	5	8	6	
6	15	13	15	15	
7	20	18	18	16	
8	17	19	19	18	
9	13	15	14	12	
10	10	13	18	15	

(a)



(b)

The Director process is responsible for opening and closing the input text file. It stores the values in a two dimensional integer array with 10 rows. You may need to use the following C functions (in addition to the necessary file & process management system calls): `fopen()`, `fscanf()`, `fseek()`, `fclose()`.

Arun's explanation of the task is rather skimpy so let me explain what I want. There are a number of ways to implement this task, but only one way to start: you need to create a file like the illustration but containing a column for each homework for each chapter. Let's set a minimum of 2 homeworks per chapter and 5 chapters. That gives you a 2-d array of 10 columns and 10 rows. The rows represent each students' grade on a homework, and the columns represent homework assignments, where all you need to know is that there are 2 per chapter. I would suggest that you create the input file sequentially, so that all the values for each homework for each chapter came before the next, and then all the children can use the same file, just starting and ending at different points (`lseek`). You could also create the input file in a spreadsheet such as

Excel where it would be easy to randomize the different rows, and then extract that as a CSV file and export it to Linux. You choose.

When the "Director" parent process creates the Manager children (=5, one for each chapter) the Director parent needs to pass the Manager children either the location of the input file and the offset at which to read it or else pass the actual values that the manager is responsible for as an array of arguments. Another possibility which is fine with me is to create a separate file for each Manager with just the quizzes for those chapters. Finally, you could choose not to overwrite the parent code but do some tricky ifs where if I am child 1 then I am manager and I actually read the parent's in memory array values and so on.

The process for creating workers is analogous. The Manager parent either gives the worker child the path of the file to open and an offset at which to read its values, or else passes the actual values to be averaged as a list of parameters, or else creates a file with just the values for which the worker is responsible.

Another way to handle this exercise is with shared memory but we aren't quite there yet.

You also have to decide how to report back and collect the averages, so that I know which chapter and homework the average grade represents.

It is essential that you provide me a written explanation of your process and how you implemented the programs. You also need to provide me with the input file with the original homework grades, and an output file with the averages, labeled in some reasonable way, for instance a spreadsheet. Ideally, you would also provide me with a script or log of execution so that we can see how much parallelism is achieved by dividing the work up among managers and workers.

Submission Instructions

- Save your input and output files, programs and explanatory text in a single folder and zip as: *task4_fullname.zip*. Make sure your programs compile and run without any errors.
- Email your code and explanatory text with subject line "Task 4 - CSc 332G - *full name*"
