Assignment #2

Due: 3/19/2012

Term Project Proposal

Due: 3/19/2012

1. Please write a one-page proposal regarding your term project for this course. The page should minimally include:
   - Title
   - Objective: describe the problem you want to solve
   - Application: potential applications of your term project
   - Expected outcome: the outputs that you expect to produce
   - Reference: list references that you found to be related to your term project
   - Milestones: dates of delivery of your project

Requirements for Assignment #2

Accounts have been created for you in our group's cluster system at 140.118.5.6:222. This system can only be accessed remotely through SSH and SFTP.

- Putty: a free open-source client for Windows
- FileZilla: a free open-source FTP client supporting SFTP.

– Exercise basic shell commands
- Manipulate files & directories

– What to do?
- Change your password (passwd)
- Make a directory named HW02 (please be exact!)
- Transfer your programs into HW02 directory
- Create **Makefile** (next week) to compile all your programs
- work on the following programs

3

Online references

- http://www.ee.surrey.ac.uk/Teaching/Unix/
  - Tutorial 1 – 4
- http://tldp.org/LDP/gs/node5.html
- http://linux.vbird.org/ (Chinese)

Assignment #2

Due: 3/19/2012

This program will 1) randomly user-specified number of spheres, and 2) compute distances between all these spheres.

You are to write programs to:

1) Get a user-specified N, D, Seed from command line arguments.
2) After specifying random seed (Seed) using srand(), write a function to generate N random spheres (3 coordinate components + radius), centers of these spheres should be in the region of [-D:D, -D:D, -D:D] and each sphere is numbered incrementally starting from 0. (of course, you need to use dynamic arrays to store all the data!)
3) Write a function to compute and store distances between all spheres, and the distances should be stored in a dynamic 2D arrays to facilitate distance queries. The 2D array may look like the figure in the next slide.
4) For each sphere, list spheres that collide with it (i.e. distances between two spheres are less than their sum of radius).

All the arrays mentioned (storing spheres & distances) should be dynamic arrays.

	0	1	2	3	4
0	0	0.989848	0.792402	0.601055	0.027588
1	0.989848	0	0.378068	0.389499	0.060154
2	0.792402	0.378068	0	0.397498	0.540462
3	0.601055	0.389499	0.397498	0	0.986639
4	0.027588	0.060154	0.540462	0.986639	0

Discuss

1. Discuss the efficiency of storing N sphere data using various data layout strategies:
   1. Option 1: double xyz[N][3], r[N];
   2. Option 2: double xyzr[N][4];
   3. Option 3: double x[N], y[N], z[N], r[N];
   4. You may also explorer other possibilities (e.g. using struct & class)
2. Discuss the efficiency of storing distances data using various data layout strategies:
   1. Simulated 2D dynamic array + row-major orientation
   2. Look-alike 2D dynamic array + row-major orientation
   3. Simulated 2D dynamic array + column-major orientation
   4. Look-alike 2D dynamic array + column-major orientation

Discuss

3. Try different compilation flags and see which one gives you the more efficient program
   - g++: -O2, -O3, -s, -march=native, …
   - icpc: -O2, -O3, -s, -xHost, -fast, …
   - You can find more possibilities through man g++ or man icpc

Note

- All discussions need to consider different program sizes (e.g. N=10, 50, 100, 200, 500, 1000, 2000, 5000, 10000)
- All discussions should be facilitated by charts, and then tell stories from these charts.
- All your work must reside in folder HW02 in your home directory in our cluster system (ssh into 140.118.5.6:222)
- All your programs must be made by simply typing make in HW02 folder (i.e. you must write a Makefile)
- For evaluating performance or efficiency, use the timing class that is provided to you:
  - /home/courses/stopWatch.h, /home/courses/stopWatch.o