University of Toronto
Department of Computer Science
CSC 228 - Winter 1995

Term Project

This term, you have a choice of term projects. You can either implement a WWW search engine or create an algorithm animation. Both projects have their respective rewards and drawbacks, and both will be marked by essentially the same criteria.

WWW Search Engine

This project involves the construction of a simple search engine, similar to those already in existence. While the system you develop is not going to have the enormous file storage typically available to the popular engines, you should consider this as only a temporary limitation, and accordingly, design your system so that it could scale up gracefully.

The main objective of your search engine should be to permit efficient retrieval of universal resource locators (URLs) based on the search criteria supplied by the user. The range of search options is completely up to your group. You must decide what are the most important features. For ideas, look at some of the more popular search engines such as:

- `http://webcrawler.com`
- `http://www.opentext.com`
- `http://lycos.cs.cmu.edu`
- `http://www2.infoseek.com`

The database on which your search engine will operate should be constructed by a separate program. It is your responsibility to select a sufficiently diverse set of URLs as input to this program. Note that you can save the entire contents of any URL to your directory by selecting the save option from the Netscape file menu. Keep in mind that file storage is not free, and since you will want to index as many URLs as possible in your limited space, you should plan to store your database efficiently.

The interface to your search engine can be a simple program of your own design, or, if you are ambitious enough to use HTML, you can design the interface as a web page. Note that marks will not be given for the appearance of the interface -- only for what it can provide as output. If you opt to create your interace as a web page, then be warned: at present, students cannot create their own cgi-bin scripts on the CDF system, but we are working on enabling you to do so for this course. For now, you should not rely on this being possible.

An important note about designing your search engine: some of the file structures and query processing techniques you may require will not be covered until after the project plan is due. Therefore, certain aspects of your initial design may undergo some major revisions at this point.

Algorithm Animation

In the last few weeks, you have seen algorithm animations used to demonstrate several of the file structure algorithms taught in CSC 228. The XTango system, installed on the CDF computers, will facilitate your task in constructing similar programs. Documentation and sample programs can be found in `~jer/228/xtango`. You should begin by reading the `README` file in that directory.

You may choose to create an animation of the master file update algorithm or the dynamic hashing method, or consult with the instructor if you wish to suggest one of your own, possibly an improved version of one of the existing animations. The key idea in creating an animation is to make the algorithm clear and understandable. Basically, your markers will be asking, “Does this animation really help me understand the algorithm better than the text book?” Be warned that creating a good animation is not as easy as it sounds. It requires technical, artistic, and presentational expertise. Pay close attention to the following advice:

Modularity is essential in creating animations, so take some time designing both the program structure and animation before you sit down at a computer. There are a least two ways to attack the problem. One way is to implement the algorithm without animation first. Then go through the algorithm and add the code to do the animation. This method will give you a better understanding of the algorithm before you attempt to animate it, but it may not be feasible for the short time you have to develop the project. The second way is to design both the algorithm and the animation simultaneously. This method helps when portions of the algorithm need to be implemented in a specific way to facilitate animation. This method can be used if you start small and develop the core routines first.

Project Plan: Due in tutorial, November 14

The plan should be a formal description of approximately three to five pages. It should report the membership of your team, the tasks undertaken by each member, and your plans for what is to be accomplished in your project. Your plans should include explicit mention of how you are going to arrange your program code, including a brief description of the function of each module.

Group Presentation: in tutorials, November 28 and/or December 5

Each group must make a brief (five to ten minute) presentation of their project, discussing their overall design strategy as well as special features of their system. You will also have the opportunity to demonstrate your system, following the group presentations. Note that every member of your group should be involved in planning the presentation, even if only one of you will be speaking to the class.

Final Project Report: Due in tutorial, December 5

This report will likely be longer than your reports for the first two assignments, although it should not exceed ten pages.[1] It will be similar in format, except that you must be careful to explain which members of your group did what tasks. Test input and output may not be as important as they were for previous assignments, but they should be included as a demonstration that your system works correctly. Your testing marks for the project will also come from the TAs' execution of your program.

One of your team members must place the executable program in a directory called `~/project`. The executable can be called whatever you wish. At the end of the term, *but not before*, make sure that the files in the project directory, as well as the directory itself, are readable and executable[1] by your other team members, otherwise, we will not be able to mark it. Put the name of this directory on the title page of your report.

Approaching the Project

There are several major components to either project and each will require attention early in the design. It will be natural as you plan your system to think about how to describe it in your report. The report should not be written at the end of the project. Rather, you should write an outline at the beginning, and continue to expand it as you proceed, until by the end of the coding and experimenting, the report is almost complete.

Try to build up the system from a simple skeleton to a fully functional package in gradual stages, testing various components as you proceed. With the confidence of having something simple working soon, you can go on to grow a bigger project successfully.

Dividing the labour will be difficult and will largely depend on how you design the system. Don't forget that writing the report is another big aspect of the project. It is not acceptable for all the coding or testing or report writing to be done by one person. Each team member must have at least some involvement in every aspect of the project.

---

1. Remember -- a carefully written, concise report is of far greater value to the reader than verbose drivel.

1. See `man chmod` for more details.