CS 377 : OS Design Projects, 2014

The class is divided into three sections for the purpose of the OS design projects.

- Theme A: Design of Virtual Memory for the Pranali OS (Group size: 6 persons)
- Theme B: Design of a File System for the Pranali OS (Group size: 6 persons)
- Theme C: Simulation of (Virtual Memory + File System) in C language (Group size: 2 persons)

The list of persons in these 3 themes will be displayed soon. No changes in themes will be permitted for persons in themes A and B. You will be permitted to form your own groups *within your theme.*

The projects have a common starting date, which is Friday 7 March 2014, and will have a common ending date toward the end of the semester. The projects will have a running time of approx 5.5 calendar weeks. During this time, there will be 6 lab sessions.

It is expected that the work will be performed primarily during scheduled lab hours. We will also have discussion sessions during the lab hours. The evaluation will consist of a demo + viva.

This writeup provides the hightlights of the three design projects. The specific deliverables and schedules might be different for each of the themes, and would be announced in due course.

Project A: Design of Virtual Memory for the Pranali OS

The core Pranali OS creates the entire address space of a process in the RAM, i.e., it does not provide virtual memory. This design project is concerned with the design and implementation of virtual memory for Pranali.

The project will consist of the following broad components. You will be expected to overlap the design and implementation of various components so as to complete the project in time.

1. Design of *Sim_disk* so that it simulates a real disk: The read/write calls on *Sim_disk* will lead to "blocking" of the process that has initiated the read/write operation. Specification of *Sim_disk* will be provided separately.
2. Constituting the swap space of a program on *Sim_disk*.
3. Design of the page table per process.
4. Handling of page faults by using an effective page replacement policy.
5. Design of advanced facilities such as an OS thread for managing the free frames of RAM.
6. Demonstration of the virtual memory by running 'real' C programs.

Project B: Design of a File System for the Pranali OS

The project will consist of the following broad components. You will be expected to overlap the design and implementation of various components so as to complete the project in time.

1. Design of *Sim_disk* so that it simulates a real disk: The read/write calls on *Sim_disk* will lead to "blocking" of the process that has initiated the read/write operation. The disk will be used to store the files and directories of the file system. Specification of *Sim_disk* will be provided separately.
2. Design of the directory structure of the file system.
3. Allocation of disk space for files.
4. Protection of files.
5. Design and implementation of advanced facilities such as a *disk cache*.
6. Demonstration of the file system using 'real' C programs. You will achieve this by providing your own C library so that file system calls made by compiled C programs actually activate the handlers of the system calls you have added in Pranali.

Project C: Simulation of (Virtual Memory + File System) in C language

(*Note:* This theme has nothing to do with Pranali OS.)

You are to simulate the operation of an OS that has a virtual memory and provides a file system. Overview of the simulation set-up is as follows:

Your main process will act as the OS. It will create two kinds of threads. Threads of the first kind, if any, will perform OS functions. Threads of the second kind will simulate user processes running in the OS. The operation of each thread will be governed by a *command file*. Thus, if we have n processes $p_1, p_2, \ldots p_n$, we will have n command files cf_1, cf_2, $\ldots$ cf_n, where operation of user process p_i will be governed by commands in the command file cf_i. The commands will simulate significant events in the operation of a process, e.g.,

(a) A 'read' reference to a specific page of the process address space.
(b) A 'write' reference to a specific page of the process address space.
(c) A 'file open/close' operation on a specific file.
(d) A 'read/write' operation on a specific file.
(e) Some other similar operation(s).

The project will run in the *assignment mode*. In each lab session, you will be expected to design and implement specific items. The project will consist of the following broad components.

1. A simulated disk *Sim_disk* for storing the swap space of processes as well as the files and directories of the file system.
2. A set-up for handling page faults and performing page replacement.
3. Design of the directory structure and allocation of disk space.
4. Protection of files.
5. Design and implementation of a unified *disk cache* for holding both pages of virtual memory and parts of files.