AOS PROJECT REPORT

Ayush Kumar Singh (07) Rishav Kohli (26)

Problem Statement

Simulation of Buffer Management System in context of UNIX Operating System.

Programming Language

C++ has been used as the programming language.

AOS Concepts Implemented

Multiprogramming has been achieved via threads. Threads have been used to simulate the processes. Threads call a function depicting the behavior of a process. Threads have been created via the use of <thread> header file.

Locking and Unlocking has been achieved via mutex locks on the critical sections i.e. the kernel data structures manipulation within the program accessed simultaneously by multiple threads depicting processes by making use of <mutex> header file available in C++. Output statements have also been locked/unlocked for user friendly interface and synchronization among the threads for printing outputs.

Signal Handling has been done via <condition_variable> and making use of wait and notify_all functions on threads along with mutex variables.

GETBLK and BRELSE Algorithm

All the cases arising in the getblk algorithm have been handled with utmost care.

Threads after creation execute a function namely process() where each thread requests for a no. of user defined block no. and request type (i.e. read or write) and on successful assignment by the getblk algorithm works on the block and then releases the block with a corresponding status (free in case of read and delayed write in case of write request) making use of brelse algorithm.

A fixed no. of buffers, four Hash Queues along with a Freelist depicting list of free buffers together consists of Buffer Cache.

Initially all the buffers are assumed to be free (and on the freelist) and the hash queues are empty.

Delayed Write of a buffer has been done asynchronously by spawning a thread to do the task.

Synchronization among the processes has been done via mutex locks, condition variables and signal handling.

Secondary Memory has been simulated via an array of integers holding data for the blocks. This helps us to depict how data is being changed by the processes in synchronization (especially in case of buffer marked for delayed write).

Note: Since threads are being used to simulate the functioning of Buffer Management nothing can be guaranteed about the flow of the program. However measures have been taken to avoid any kind of error or exception arising out of the program.

Team Members and respective Contributions

Team Member 1: Ayush Kumar Singh (07)

All the thread related tasks along with process simulation and signal handling has been done by him.

I learnt how to make threads work in synchronization making use of signal handling and working with shared data structures in multithreaded environment.

Team Member 2: Rishav Kohli (26)

The creation of Buffer Cache and methods to manipulate and work with it have been done by him.

I learnt how to make effective use of data structures and creating methods in such way that are easy to work with in a multithreaded and synchronized environment.