**CSE 150 Operating Systems**

**Design Phase 2: Multiprogramming**



**Authors: Team Thimble**

Andrew Yfantis

Isaac Flores

Landon Taylor

Lenny Khodel

**TASK I:**

Implement the file system calls: creat,open,read,write,close, and unlink. The methods are under nachos.userprog.UserProcess.java. The file system has already been implemented for us, and we will simply allow user processes to access said file system. Syscall.h is under nachos.test, which is the documentation for the file system calls.

We must bullet-proof the Kernel from user program errors, meaning all cases that would corrupt the internal state of the kernel or that of other processes. This can be done by…

To fix halt() so that it can only be invoked by a root process, we will need to check if the process is a USERKERNEL class. We will probably do this by adding an extra variable that is set inside of the default constructor of the UserKernel class.

Class UserProcess

{

Private int handleHalt(){

If(thisprocess!= root process)

Return 0

Halt

assertNotReached(“halt didn’t work”)

return -1;

}

}

**TASK II:**

Implement support for multiprogramming. Allocate physical memory so that processes do not overlap. There are no dynamic memory allocation needs. Assuming 8 pages for a processes stack is okay.

**TASK III:**

Implement system calls: exec, join, exit.

**TASK IV:**

Implement a lottery scheduler. The major difference is the mechanism used to pick a thread from a queue. Lottery scheduler must implement priority donation. Instead of donating priority, waiting threads transfer tickets to threads they wait for.

The owner’s ticket count Is the sum of its own tickets and the tickets of all waiters, not the max.

Do not keep an array containing an entry for every ticket.

Real tickets in the system is guaranteed not to exceed Integer.MAX\_VALUE, instead of 7.