CSE 150 Project #2 Group #3

Group Members: Ryan Hules, Yiying Jie, Chi Hong Kou, Haoxian(Andy) Huang, Jonathan Huang, Nathan Huizar

Task I: File System

Goal: Implementing the file system calls - creat/e, open, read, write, close and unlink, documented in syscall.h. *Not* implementing a file system, but give user processes the ability to access a file system.

1. Creat/e
    a. Search within memory the input fileAddress
    b. See if the file already exists, that is, if the fileAddress is null
    c. If the file does not exist, create a new file
        i. Find the appropriate space empty space
        ii. Set boolean create to TRUE to create file
    d. Put in the fileDescriptor into file that was just created

Pseudocode:

```
openfile[fileDescriptor] =ThreadedKernel.fileSystem.open (filename, false);
if filename is null
        return -1; // create failed
// else…
int fileDescriptor = findEmpty();
if fileDescriptor != -1
        Create new file
else
        return -1; // create failed
```

2. Open
    a. Read the file from virtual memory according to user input fileAddress
    b. Have to make sure fileAddress is valid, implement check
    c. Open file, and make sure the value of create is FALSE
    d. Put in fileDescriptor
3. Read

   a. Implement condition checks if greater than 16 or less than 0 or null
   b. Read file and save in temporary memory, return
   c. Write the content to the virtual memory address, return numbers that was successfully written

4. Write
   a. Gives address of the write file, address, bytes of writing memory
   b. Implement condition checks if greater than 16 or less than 0 or open returns null/error (same as read)
   c. Read file into temporary memory, if error, return
   d. Write the contents to the disk, return
   e. Implement other condition checks, such as if successfully written data is greater than data to be stored, return
5. Close
   a. Implement condition checks, if greater than 16 or less than 0 or null
   b. Close open file, set the address to null
6. Unlink
   a. Remove data according to the address sent in
   b. Implement condition to check, if the file does not exist, no need to remove anything

Pseudocode:

```
if fileName is null
        return; // no need to unlink/delete
remove file, return;
```

Task II: Multiprocessing

Implement support for multiprogramming. The code given is restricted to running one user process at a time. We need to make it work for multiple user processes.

- Add a variable to maintain list of free physical pages, as well as adding a free page into the list.
- Variable to remove first element of page list and return the number of free page.
- Modify **UserProcess.readVirtualMemory** and **UserProcess.writeVirtualMemory**, which copy data between the kernel and the user's virtual address space, to work with multiple user processes.
- Modify **UserProcess.loadSections()**so that it allocates the number of pages that it needs
  - This method should also set up the **pageTable** structure for the process so that the process is loaded into the correct physical memory pages.

- The physical memory of the MIPS machine is accessed through **Machine.processor().getMemory();** the total number of physical pages is **Machine.processor().getNumPhysPages()**.
- Implement **UserProcess.unloadSections()** to release page tables and back out physical pages

Pseudocode:// All current code given

```
public int writeVirtualMemory(int vaddr, byte[] data, int offset,int length) {
        Lib.assertTrue(offset >= 0 && length >= 0 && offset+length <= data.length);
        byte[] memory = Machine.processor().getMemory();
        // for now, just assume that virtual addresses equal physical addresses
        if (vaddr < 0 || vaddr >= memory.length)
           return 0;
        int amount = Math.min(length, memory.length-vaddr);
        System.arraycopy(data, offset, memory, vaddr, amount);
        return amount;
   }
```

Task III: System Calls

Exec

- Create new child process
- Execute program stored in specified file using said new process
- Return -1 on error

Join

- Set up process ID counter (assume it will not overflow)
- Add thread parameters so children know who their parents are
- isChild() helper function?
- Check to ensure that threads can only be joined by their respective parent
- On call, check exit status of specified child process
- Sleep caller until said child process exits
- Disown child process on exit

Exit

- Close open files
- Free Memory
- Disown any children

- Pass exit status to parent
- If last process, call Kernel.kernel.terminate()
- Terminate thread

Task IV: Lottery Scheduler

- Priority donation
- Extend priority scheduler
- Waiting thread must transfer tickets
- Do not use array
- increase/Decrease should +1 and -1
- Min and Max values have changed

```
Public lotteryScheduler {

Public static final int priorityDefault = 1;
Public static final int priorityMaximum = Integer.MAX_VALUE;;
Public static final int priorityMinimum = 1;


        //will have its own set priority func that will pass in the thread and its associated priority
        Public void setPriorirty( KThread, priorty){

        //check if the priority is less than priorityMax
        //check if the priority is greater than priorityMin

        }

        //make a priortyQueue for the lotteryQueue
        Protected class lotteryQueue extends PriorityQueue {

        //here we will get the transfer priority and randomize

        //we will also add an effectivePriority function
        Public int getEffectivePriority(){

        //here we will check if the current threads priority changed and make
        //sure the correct priority was given

        }

        //we also make sure that we do this for the threads after the current one
        Public ThreadState pickNextThread(){
```

```
        //here we will work out assigning the winning tickets
        //and handle effective priorities to each respective thread

        }

        }

}

//a lot of the previous code we constructed for Priority Scheduler will be used to simplify the
//functions we are making for our new Lottery Scheduler
```