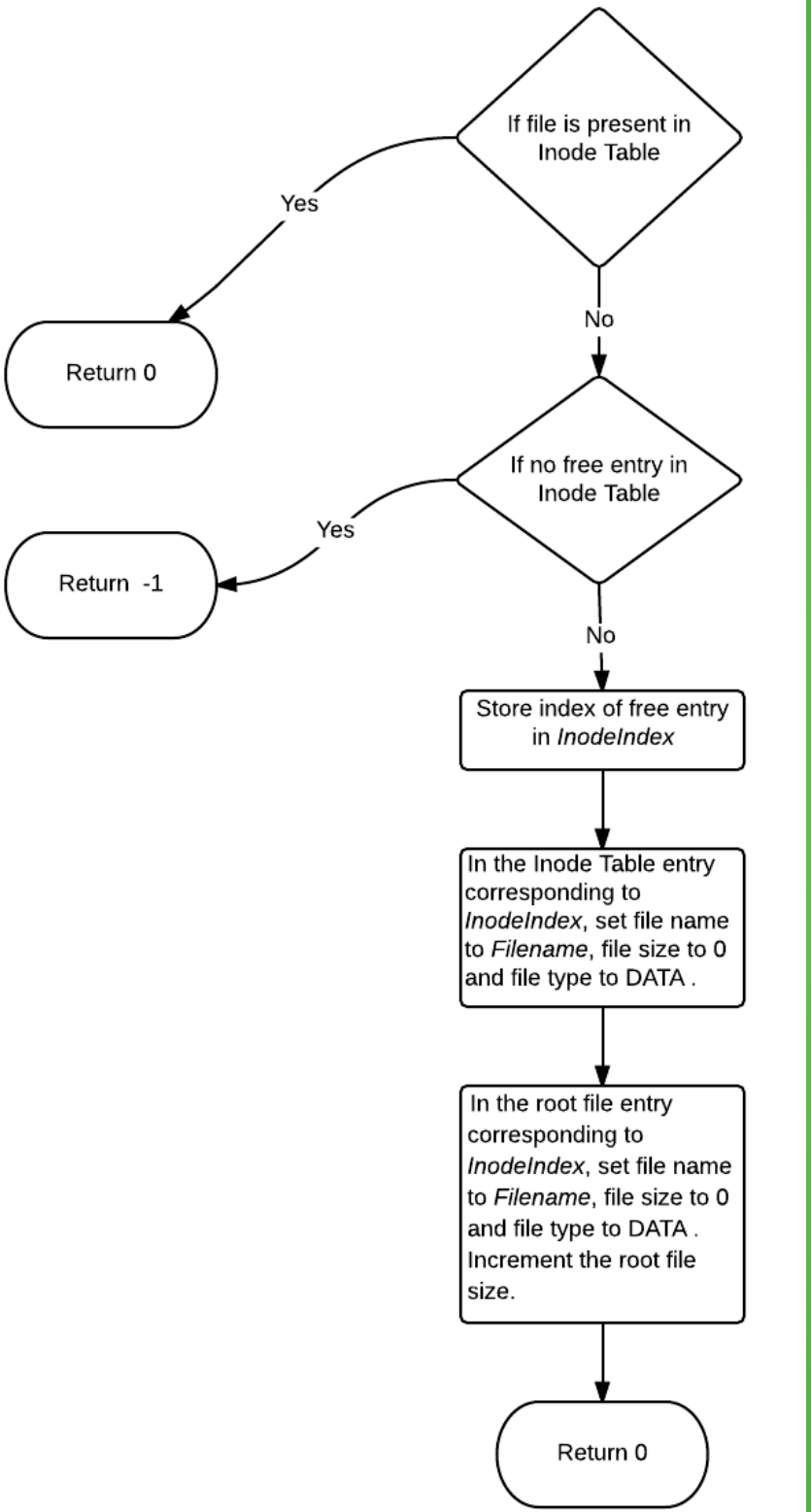
Enhancement to eXpOS Operating System and eXpFS File System

- Kruthika Suresh Ved
Sikha V Manoj
Sonia V Mathew
Guided by: Dr.K.Muralikrishnan

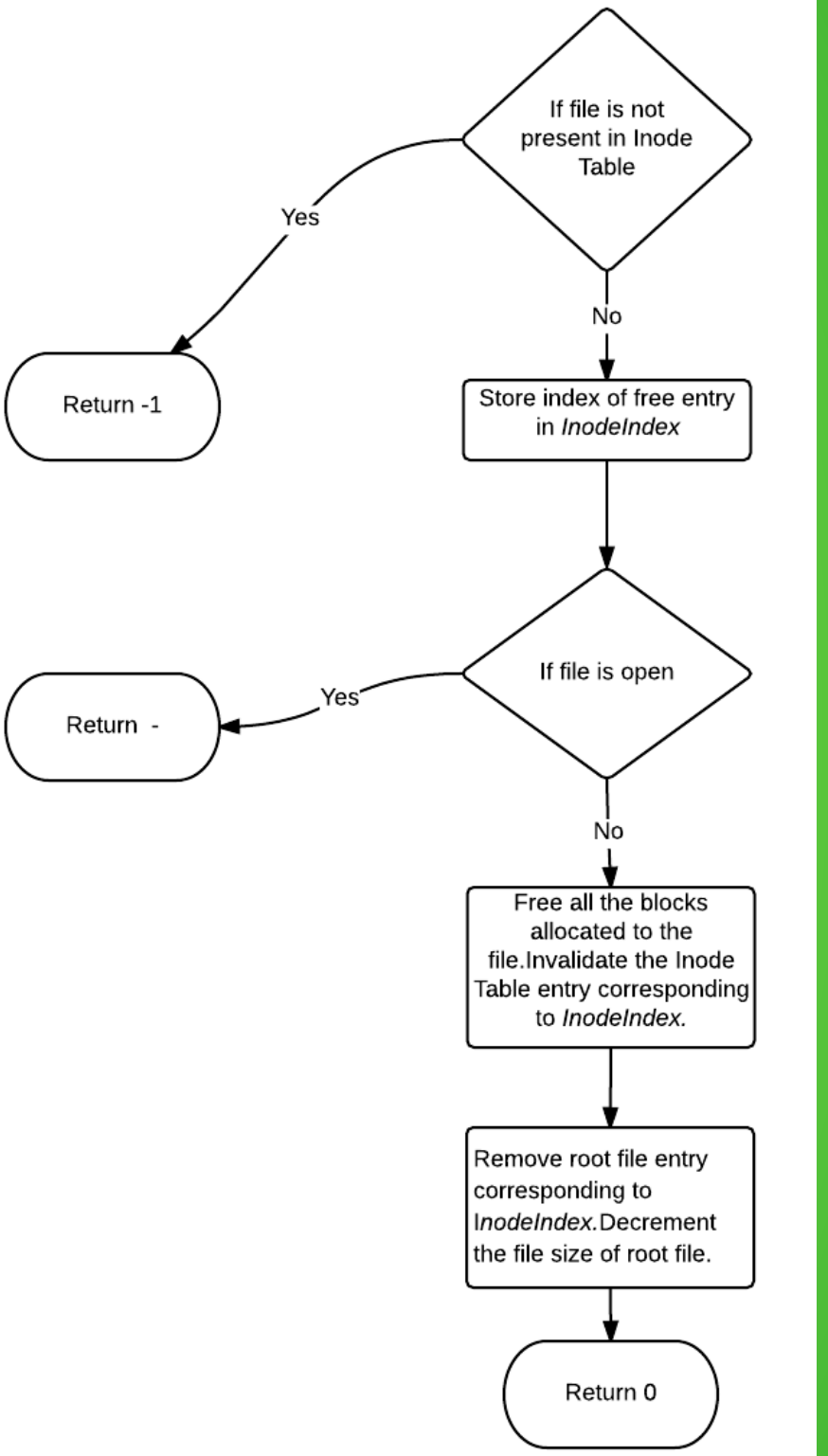
File System Calls

- Create
- Delete
- Open
- Close
- Read
- Write
- Seek

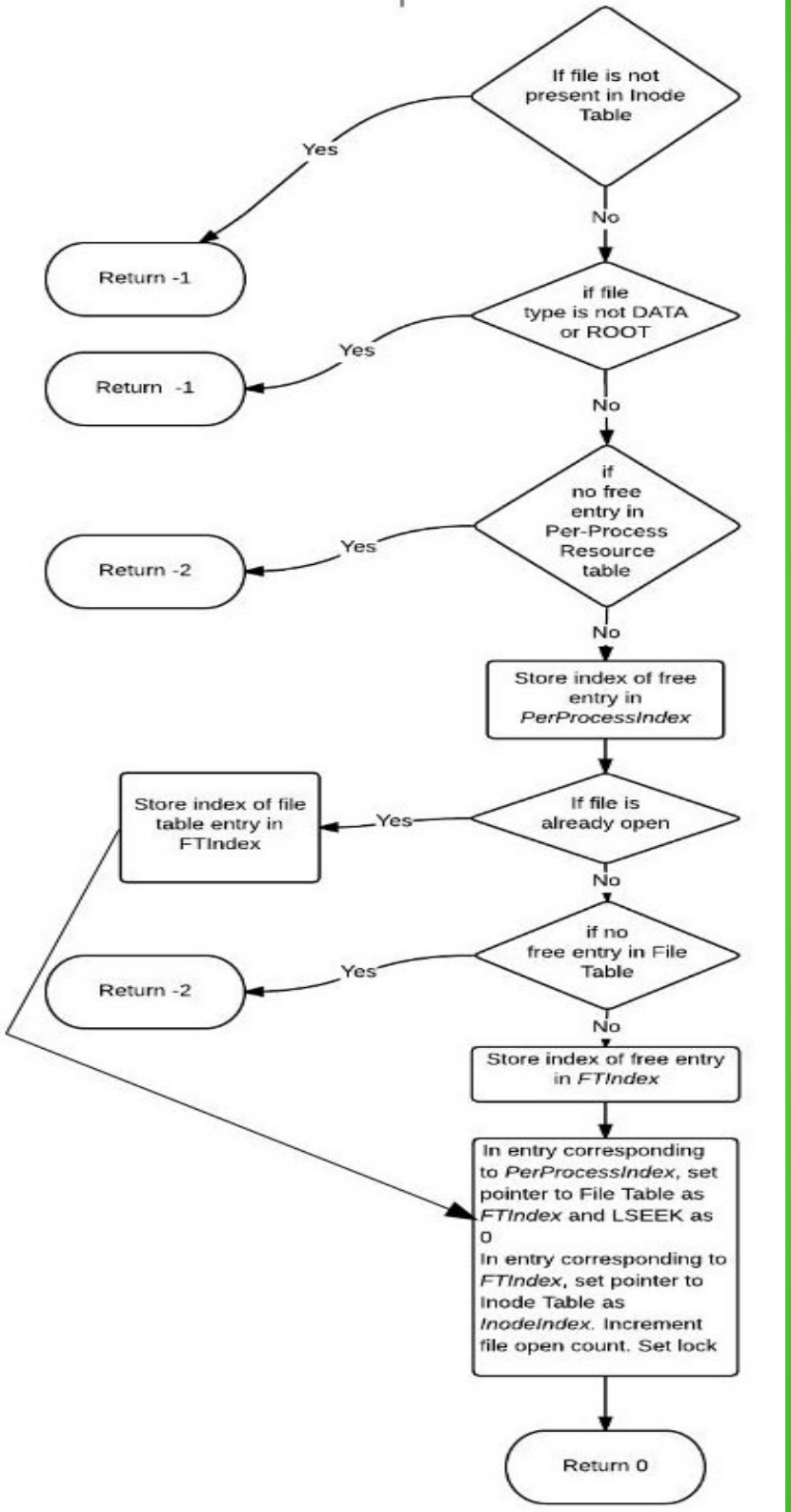
Create System Call



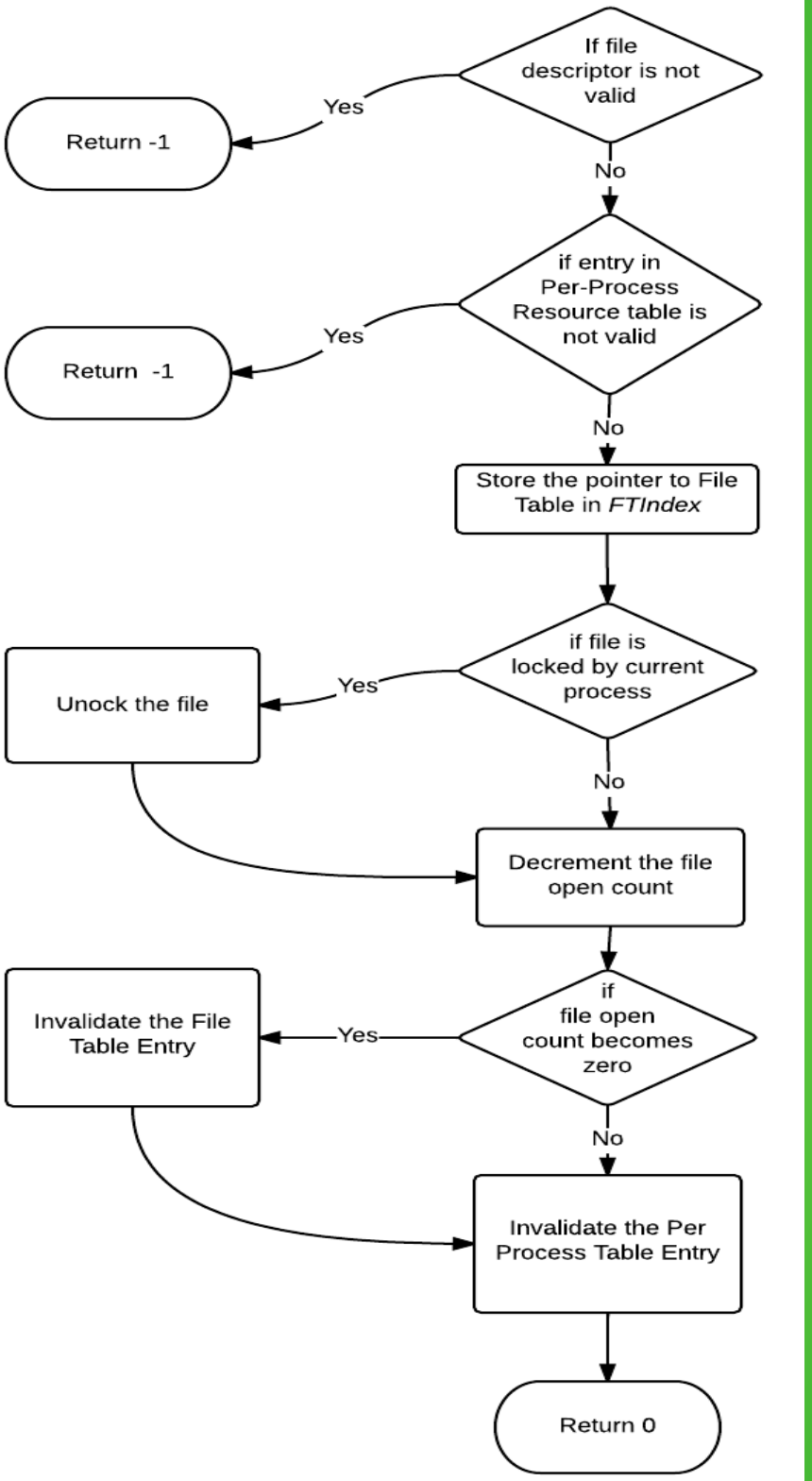
Delete System Call



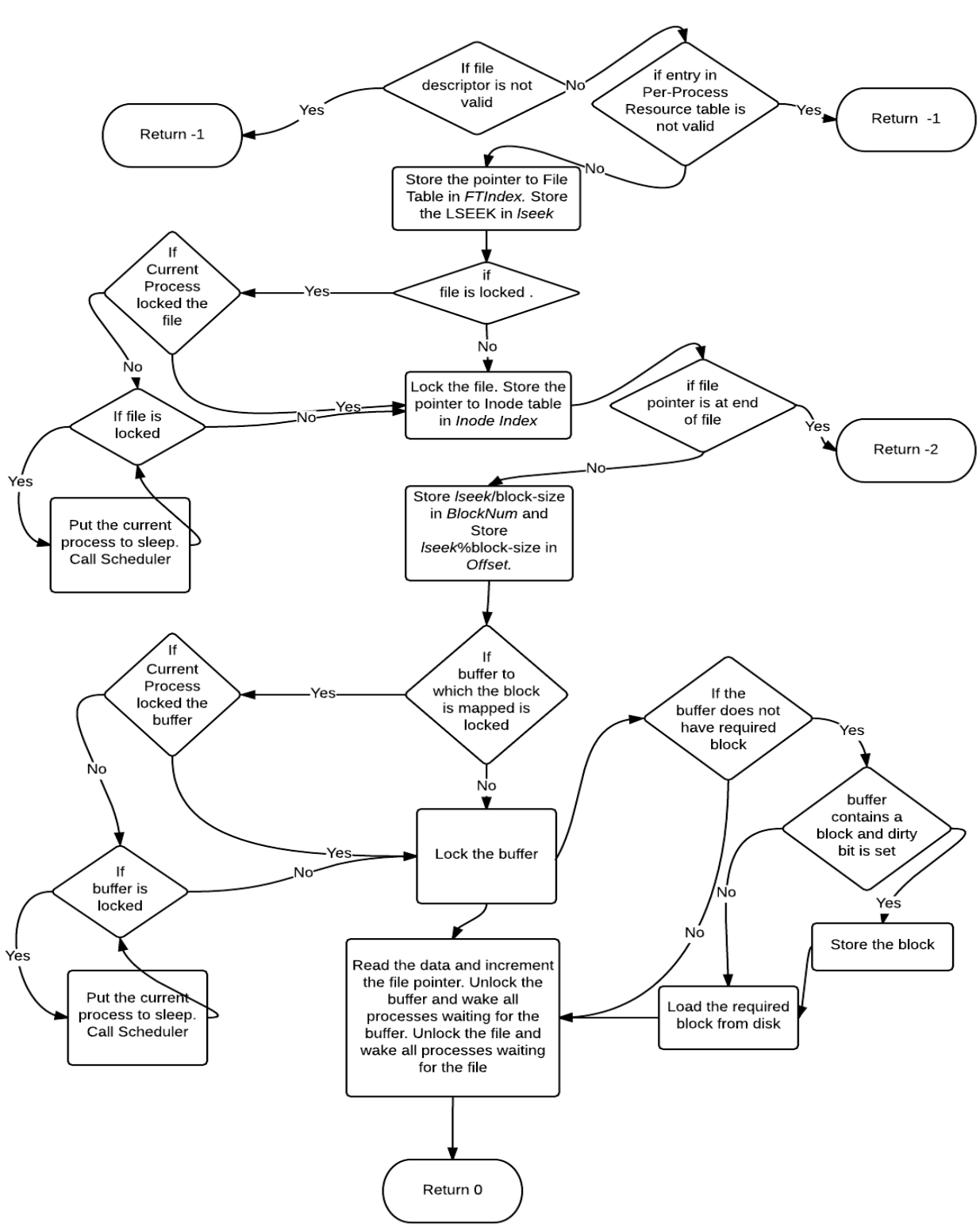
Open System Call



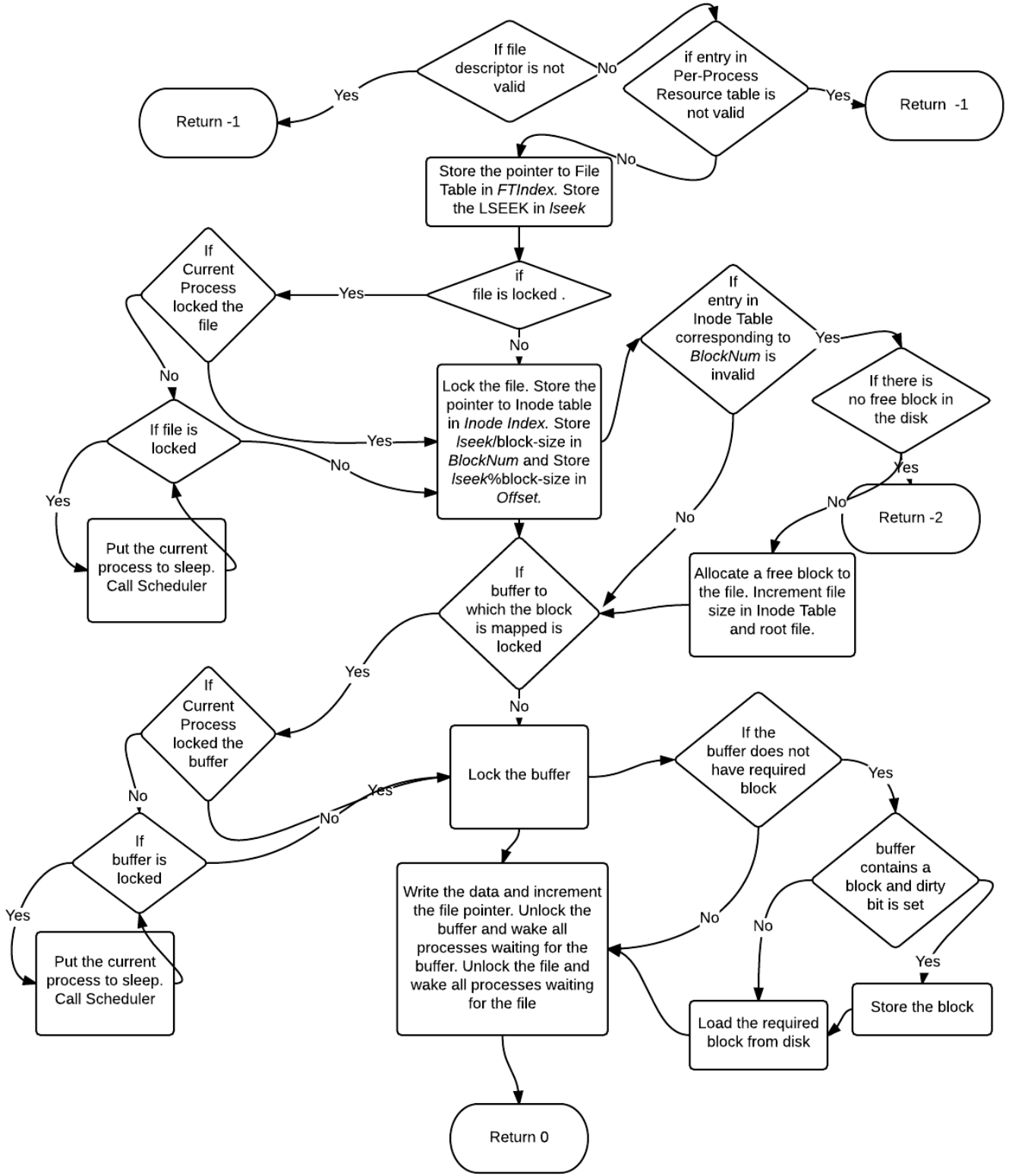
Close System Call



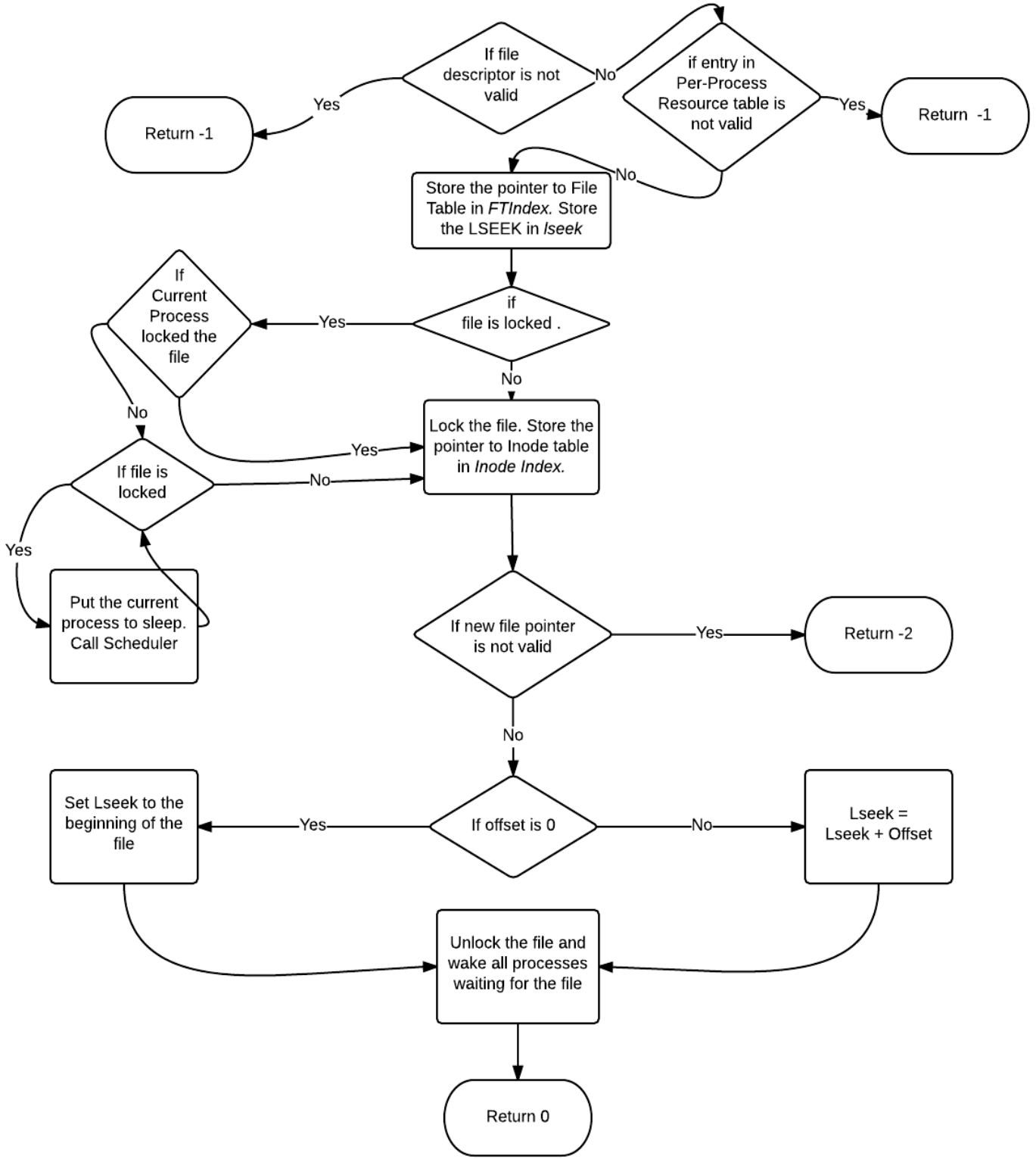
Read System Call



Write System Call



Seek System Call



Process System Calls

- Fork
- Exec
- Exit
- GetPid
- GetPPid
- Shutdown

Algorithm 8 Fork system call

```
if no free entry in process table then
    return -1
else
    Store index of free entry in ChildPID
    Store pid of parent process in ParentPID
end if
Set the PPID field of child process to ParentPID
Count the number of stack pages of parent
while equal number of free pages are not present in memory do
    Put the process to sleep
    Call scheduler
end while
Allocate one free page to the child for each stack page of the parent
Copy the parent's stack to child's stack
Copy the page table entries, except stack entries, of parent to the page table of child
Copy the parent's machine state and Per-Process resource table to the child
Copy the inode index from parent to child
For every open file of the parent, increment the file open count
For every semaphore acquired by the parent, increment process count
Set state of child to ready
return 0 to the child process and ChildPID to the parent process
```

Algorithm 9 2. Exec system call

if file not found in Inode Table **then**
  return -1
else
  if file type is not EXEC **then**
    return -1
  else
    Store index of Inode Table entry in *InodeIndex*
    Store the code block numbers of the file in *Block1* and *Block2*.
  end if
end if
In the page table of current process, set code page entries to *Block1* and *Block2*.
Set the auxiliary information of code pages to invalid and unreferenced.
Include the page numbers of shared library in the page table.
Invalidate the entry for heap pages
In the process table of current process, set the pointer to Inode Table as *InodeIndex*
Close all files opened by the current process
Release all semaphores held by the current process.
Set SP and IP values to valid locations.
return 0

Algorithm 10 Exit system call

if no more processes to schedule **then**
  Shutdown the machine
else
  Store the pid of the next ready process in *NextPID*
end if
Close all files opened by the current process
Release all the semaphores used by the current process
Memory pages of the current process are freed
Invalidate the page table entry
Wake up all processes waiting for the current process
Schedule the process with pid *NextPID*
return

Algorithm 11 Getpid system call

Find the PID of the current process by using PTBR value.

return PID of current process

Algorithm 12 Getppid system call

Find the PID of the current process by using PTBR value.

From the Process Table entry of the current process, find the PPID

return PPID of current process

Algorithm 13 Shutdown system call

while disk is not free **do**

  Put the process to sleep

  Call scheduler

end while

Store Inode Table to the disk

Store dirty pages to disk

Store Disk Free List to the disk

Halt the machine

return

System calls for access control and synchronization

- Wait
- Signal
- Flock
- FunLock
- Semget
- Semrelease
- SemLock
- SemUnLock