**CIS 657: Principles of Operating Systems**

**Lab1 TasksCB**

- Bharath Darapu

- Avinash Peddi

In this lab project we are assigned to finish the following implementations in the TasksCB.java file:

1. do\_create (TaskCB) and

2. do\_kill (TaskCB).

The main idea in this module is dealing with the creation and killing of tasks (which holds as the container for threads).

**Approach used:**

The approach used is quite simple and is mostly straight forward. Let me explain the implementation in the order of occurrence in the program.

At the beginning of the class we have declared a few private variables as below:

***int*** *pid; /\*process id\*/*

*/\*PageTable clear variable for storing page table information\*/*

***static*** *PageTable pPageTable;*

***private*** *OpenFile pSwapFile =* ***null****;*

*/\*GenericList class variable declaration\*/*

***private*** *GenericList pThreadList; /\*will contain threads\*/*

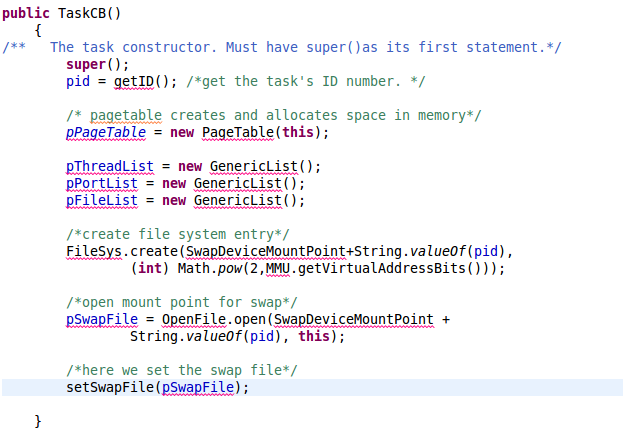
***private*** *GenericList pPortList; /\*will be used for ports\*/*

***private*** *GenericList pFileList; /\*will hold the file list\*/*

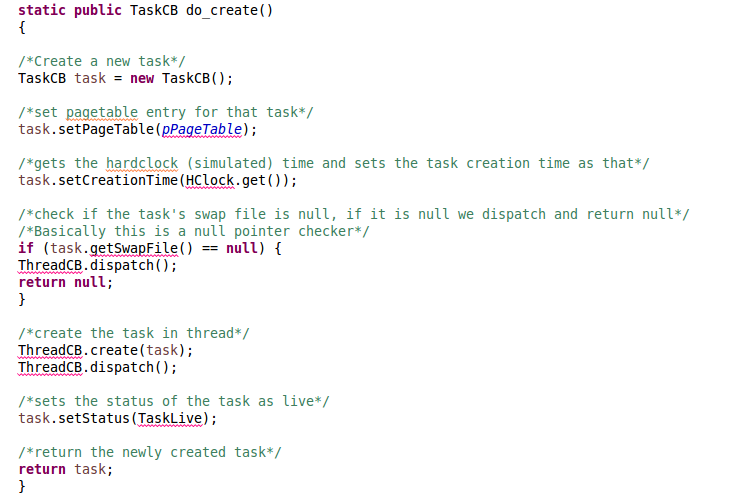
- The pid will be used to store the task's id so as to be used later to add or delete the task from the file system.  
- The page table variable was created and will be used later for allocation or de-allocation of memory.

- Swapfile variable will be used to open and close a mount point for the task which we will create later.

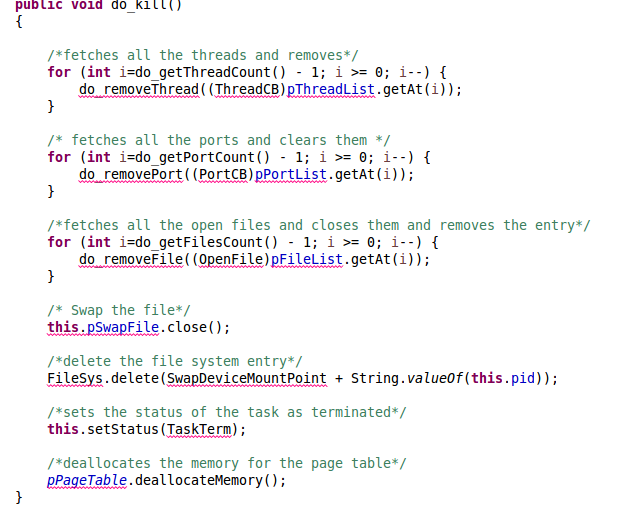
- Three lists are being used for holding the three lists of open threads, open ports and open files.



* The TaskCB constructor is being used to get the task's id number (and assign it to pid), create memory for the page table, create the lists for threads, ports and files.
* Then once we get the pid, we create a file system entry using the filesys.create() method call and as arguments pass the pid and the calculated memory (size equal to the size (in bytes) of the addressable virtual memory) to be used.
* Next we create a mount point for the task. And set the swap file.



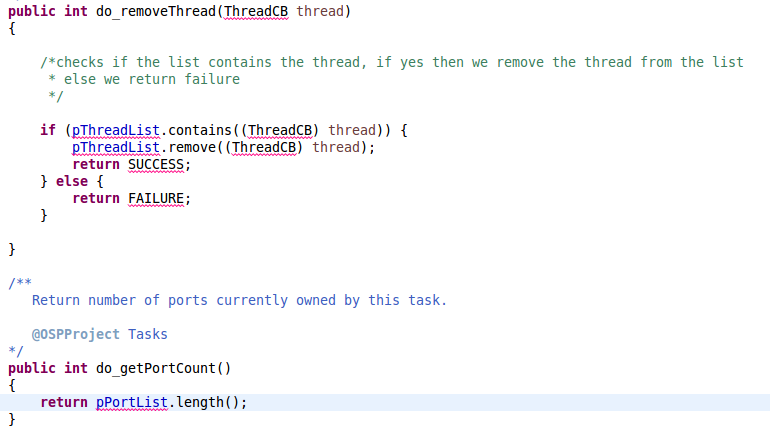
* This is one of the critical methods. In this do\_create method we create a new thread list, set TaskLive status and creation time, create and open the task's swap file.
* New task is created using 'TaskCB task = **new** TaskCB();'
* Then we set the page table which we created earlier and then set the creation time we get from the hardclock.
* Then we create and open the task's swap file and set the status of the task as 'live'



* Next is the do\_kill method, here we kill the specified task and all of it threads.
* Set the status, free all memory frames and delete the task's swap file.
* First we fetch all the threads of the related task and then remove them one after the other, with in a for loop. Same applies for the ports and open files.
* Then we close the swap file and free the memory frames by using the filesys.delete and passing the pid.
* Next we set the status of the task as terminated.
* And finally we de-allocate the memory.

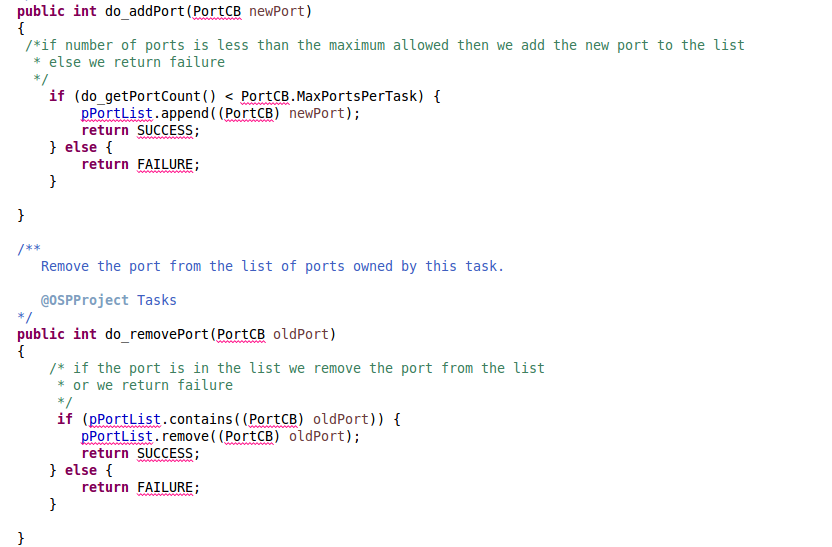


* ‘pThreadList.length()' is used to fetch the number of related threads.
* **'**pFileList.length()' is used to fetch the number of open files.
* Next in the add\_Thread() method we will check if the thread count is under the maximum permitted limit per task, if it is then we add the the thread (passed) to the thread list for that task. Else we return a failure signal.



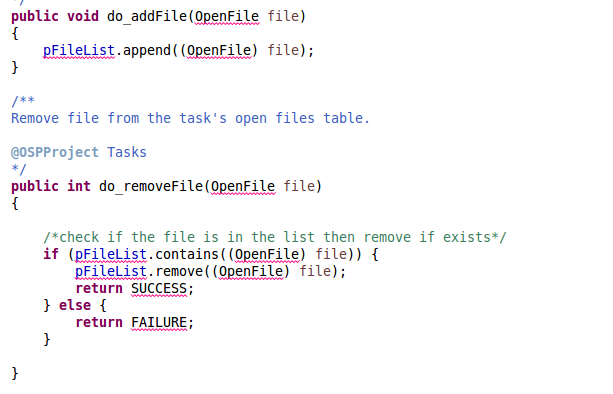
- In the do\_removeThread method() we will check if the passed thread is actually in the task's thread list. If it is then we remove the thread from the list or we return a failure.

- 'pPortList.length()' is used to get the number of ports.



- In the do\_addPort metthod a new port is passed and if the number of ports currently is less that the maximum number of ports allowed then we add it to the list.

- In the do\_removePort method if the port is in the existing port's list then we remove it or we return a Failure.



- In the do\_addFile method 'pFileList.append((OpenFile) file)' we append the file to the file list.

- In the do\_removeFile() method we check if the file (passed) is in the file list then we remove It and remove a success or we return a failure.

After all the implementation we navigated to the project path and

* Compiled the .java file using ‘javac –classpath .;OSP.jar \*.java’ and
* Later used ‘java -classpath .;Demo.jar osp.OSP’ command to run the simulation.

OSP.log file which was generated during the execution is also attached.