## Task 3: Implement System Calls

### Introduction

When implementing System Calls, we use these system calls:

Join() is a system call that grabs process ID from the arguments. When using join, you can only use it with one of its child processes

exec() is a system call that creates and executes a user's process. It uses 3 parameters where the first one must have a .coff in the filename in order to execute it. The second parameter is the number of arguments and the last parameter is a pointer for the arguments.

exit() is a system call that terminates the user's process.

Pseudocode - join()

```
public int join(int processID, int status) {
// creates child
UserProcess child = NULL;
//Finding the child process
        for(int i = 0; i < childProcessID.size(); i++){
        //  if(childProcessID == process ID)
        // then return child = childProcessID
        }
//Needs to check if childProcess is valid
        if  (child == NULL) {
        //if invalid you want to exit so that method is called
         }
// Join the Child Process to Parent Process
        child.thread.join();
// Needs to check if processID and status are valid
        if(processID < 0 || status < 0) {
        //if invalid then exit
        }
if(processID != Null && status != Null) {
            //Check the Child Process status
            }
}// end of Join method
```

Pseudocode - exec()

```
public int exec(int file, int argc, int argv){
// Checking if file is valid
            if (file  < 0 || argc < 0  || argv < 0) { return -1; }
//  Checking if filename is valid
String filename = readVirtualMemoryString(file, 256);
            if (filename invalid) { return -1;}
            if(!filename.contains(".coff")) {return  -1}
// Checking the arguments
if (argc < 0)  {return -1; }
//  read the  VirtualMemoryString
// constructor to execute the Child Process
}// end of  exec() method
```

Pseudocode - exit()

```
public void exit()
{
//terminate thread
//close opened files
//free up memory
//check if root is calling [if is then invoke halt()]
//check if parent has recalled join()
Machine.halt();
}
```

**Test Cases**

Test Case 1: With an invalid childcase, the program would go into the join function and would realize that the parameters are not met in order and go to the exit function where it would terminate the threads and go through the process of ending the program.

Test Case 2: If the program ends abnormally, some aspects of the exit method would not work as intended however there are some checks for how the exit is being called and why it is being called which means that errors will be caught.

Test Case 3: With a fully valid child case, the program will join and go through the process of executing but if the file is invalid then once again like the earlier test case it will be ended earlier and conclude.