Kernel.c

```
code Kernel

  --AMANDEEP KAUR
---------------------------InitFirstProcess--------------------

       function InitFirstProcess()
          var
             pt:ptr to Thread
           pt=threadManager.GetANewThread()            --get new thread
           pt.Init("UserProcess")                      --initialize new thread
           pt.Fork(StartUserProcess,0)                 --fork thread
       endFunction

-----------------------StartuserProcess-------------------

       function StartUserProcess()

         var
          initPC:int
          oldIntStat:int
          initUserStackTop:int
          addrSpace: ptr to AddrSpace
          op:ptr to OpenFile
          initSystemStackTop:int
          p:ptr to ProcessControlBlock

         p=processManager.GetANewProcess()      --get a new process/allocate new PCB
         p.myThread=currentThread               --initialize mythread field
         currentThread.myProcess=p              --init myprocess in current thread

         addrSpace = &p.addrSpace                   --create logical address space

         op=fileManager.Open("TestProgram1")      --open file
         initPC=op.LoadExecutable(addrSpace)      --load executable in it
         fileManager.Close(op)                    --close file
         initUserStackTop=addrSpace.numberOfPages*PAGE_SIZE
                                               --compute initial value of user level stack
         initSystemStackTop=(&currentThread.systemStack[SYSTEM_STACK_SIZE -1])
asInteger                                                --initialize system stack top
         oldIntStat=SetInterruptsTo(DISABLED)      --disable interrupt
         addrSpace.SetToThisPageTable()            --initialize page table register
         currentThread.isUserThread = true         --set isUserthread var
         BecomeUserThread(initUserStackTop,initPC,initSystemStackTop)
                                                 --change mode bits and perform jump

       endFunction
-----------------------------  Handle_Sys_Exit  ---------------------------------

  function Handle_Sys_Exit (returnStatus: int)
      print("Handle_Sys_Exit invoked!")
      nl()
      print("return status= ")
      printInt(returnStatus)
      nl()
    endFunction

-----------------------------  Handle_Sys_Shutdown  -------------------------------
--

  function Handle_Sys_Shutdown ()
      FatalError ("Syscall 'shutdown' was invoked by a user thread")
    endFunction

-----------------------------  Handle_Sys_Yield  ---------------------------------

  function Handle_Sys_Yield ()
       print("Handle_Sys_Yield invoked!")
      nl()
    endFunction

-----------------------------  Handle_Sys_Fork  ---------------------------------

  function Handle_Sys_Fork () returns int
       print("Handle_Sys_Fork invoked!")
      nl()
      return 1000
    endFunction

----------------------------  Handle_Sys_Join  --------------------------------

  function Handle_Sys_Join (processID: int) returns int
       print("Handle_Sys_Join invoked!")
      nl()
      print("processID=")
      printInt(processID)
      return 2000
    endFunction
----------------------------  Handle_Sys_Exec  --------------------------------

  function Handle_Sys_Exec (filename: ptr to array of char) returns int
      var
          sb: array [MAX_STRING_SIZE] of char
          initPC:int
          oldIntStat:int
          initUserStackTop:int
          newAddrSpace: AddrSpace = new AddrSpace
          op:ptr to OpenFile
          initSystemStackTop: ptr to int
          i:int

         newAddrSpace.Init()                         --initialize new address space

         i = currentThread.myProcess.addrSpace.GetStringFromVirtual(&sb, filename
asInteger, MAX_STRING_SIZE)
             --translate file pointer that is virtual address into physical address
         if(i<0)                               --check if there is no match than return
             return -1
         endIf

         op = fileManager.Open(&sb)                                      --open the file
         if(op == null)                      --check if file not able to open the file
           return -1
         endIf

         initPC=op.LoadExecutable(&newAddrSpace)        --load the executable to PC
         if(initPC == -1)                               --check if it doesnot fetch
           --Print("Loadexecutable failed")
           return -1
         endIf
         fileManager.Close(op)                                              --close file
         frameManager.ReturnAllFrames(&currentThread.myProcess.addrSpace)
-                                   --return frames allocated for previous addr space
         currentThread.myProcess.addrSpace = newAddrSpace
         initUserStackTop= newAddrSpace.numberOfPages * PAGE_SIZE
-                                                   --initialize user level stack
         initSystemStackTop = & currentThread.systemStack[SYSTEM_STACK_SIZE -1]
-                                                 --initialize system level stack
         oldIntStat=SetInterruptsTo(DISABLED)
-                                                            --disable interrupts
         newAddrSpace.SetToThisPageTable()
-                                                        --initialize page table
         currentThread.isUserThread = true
-                                                         --set isUserThread var
         BecomeUserThread(initUserStackTop,initPC,initSystemStackTop asInteger)
-                                           --change mode bits and perform jump
     return 3000
    endFunction

----------------------------  Handle_Sys_Create  ---------------------------------

  function Handle_Sys_Create (filename: ptr to array of char) returns int
      var
        sbuff:array [MAX_STRING_SIZE] of char
        a:int
        a=currentThread.myProcess.addrSpace.GetStringFromVirtual(&sbuff,filename
asInteger,MAX_STRING_SIZE)
        print("Handle_Sys_Create invoked!")
        nl()
        print("virtual address of filename= ")
        printHex(filename asInteger)
        nl()
        if(a<0)
          return -1
        endIf
        print("filename= ")
        print(&sbuff)
        nl()
        return 4000
    endFunction

-----------------------------  Handle_Sys_Open  ---------------------------------

  function Handle_Sys_Open (filename: ptr to array of char) returns int
       var
        sbuff:array [MAX_STRING_SIZE] of char
        a:int
        a=currentThread.myProcess.addrSpace.GetStringFromVirtual(&sbuff,filename
asInteger,MAX_STRING_SIZE)
        print("Handle_Sys_Open invoked!")
        nl()
        print("virt addr of filename= ")
        printHex(filename asInteger)
        nl()
        if(a<0)
          return -1
        endIf
        print("filename= ")
        print(&sbuff)
        nl()
      return 5000
   endFunction
-----------------------------  Handle_Sys_Read  ---------------------------------

  function Handle_Sys_Read (fileDesc: int, buffer: ptr to char, sizeInBytes: int)
returns int
       print("Handle_Sys_Read invoked!")
       nl()
       print("fileDesc= ")
       printInt(fileDesc)
       nl()
       print("virt addr of buffer= ")
       printHex(buffer asInteger)
       nl()
       print("sizeInBytes= ")
       printInt(sizeInBytes)
       nl()
      return 6000
    endFunction

-----------------------------  Handle_Sys_Write  ---------------------------------

  function Handle_Sys_Write (fileDesc: int, buffer: ptr to char, sizeInBytes: int)
returns int
       print("Handle_Sys_Write invoked!")
       nl()
       print("fileDesc= ")
       printInt(fileDesc)
       nl()
       print("virt addr of buffer= ")
       printHex(buffer asInteger)
       nl()
       print("sizeInBytes= ")
       printInt(sizeInBytes)
       nl()
      return 7000
    endFunction
-----------------------------  Handle_Sys_Seek  ---------------------------------
  function Handle_Sys_Seek (fileDesc: int, newCurrentPos: int) returns int
       print("Handle_Sys_Seek invoked!")
       nl()
       print("fileDesc= ")
       printInt(fileDesc)
       nl()
       print("newCurrentPos= ")
       printInt(newCurrentPos)
       nl()
      return 8000
    endFunction
```

```
-----------------------------  Handle_Sys_Close  ---------------------------------

  function Handle_Sys_Close (fileDesc: int)
        print("Handle_Sys_Close invoked!")
       nl()
       print("fileDesc= ")
       printInt(fileDesc)
       nl()
 endFunction
```