Kernel.h

```
header Kernel

  uses System, List, BitMap

  const

    SYSTEM_STACK_SIZE = 1000        -- in words
    STACK_SENTINEL = 0x24242424     -- in ASCII, this is "$$$$"

    -- The kernel code will load into the first megabyte of physical memory.  This
    -- should be more than enough.  We will use the second megabyte for page
frames.
    -- Thus, the frame region is 128 page frames of 8K each.

    PAGE_SIZE = 8192                                       -- in hex: 0x0000 2000
    PHYSICAL_ADDRESS_OF_FIRST_PAGE_FRAME = 1048576   -- in hex: 0x0010 0000
    --NUMBER_OF_PHYSICAL_PAGE_FRAMES = 512              -- in hex: 0x0000 0200
    NUMBER_OF_PHYSICAL_PAGE_FRAMES = 27               -- for testing only

    MAX_NUMBER_OF_PROCESSES = 10
    MAX_STRING_SIZE = 20
    MAX_PAGES_PER_VIRT_SPACE = 20
    MAX_FILES_PER_PROCESS = 10
    MAX_NUMBER_OF_FILE_CONTROL_BLOCKS = 10
    MAX_NUMBER_OF_OPEN_FILES = 10
    USER_STACK_SIZE_IN_PAGES = 1
    NUMBER_OF_ENVIRONMENT_PAGES = 0

    SERIAL_GET_BUFFER_SIZE = 10
    SERIAL_PUT_BUFFER_SIZE = 10

  enum JUST_CREATED, READY, RUNNING, BLOCKED, UNUSED      -- Thread status
  enum ENABLED, DISABLED                                  -- Interrupt status
  enum FILE, TERMINAL, PIPE                                -- Kinds of OpenFile

  -- Syscall code numbers for kernel interface routines
  -- NOTE: These codes must exactly match an identical enum in UserSystem.h
  enum SYSCALL_EXIT = 1,
       SYSCALL_SHUTDOWN,
       SYSCALL_YIELD,
       SYSCALL_FORK,
       SYSCALL_JOIN,
       SYSCALL_EXEC,
       SYSCALL_CREATE,
       SYSCALL_OPEN,
       SYSCALL_READ,
       SYSCALL_WRITE,
       SYSCALL_SEEK,
       SYSCALL_CLOSE
  enum
ACTIVE, ZOMBIE, FREE      -- Status of a ProcessControlBlock

  var
    readyList: List [Thread]
    currentThread: ptr to Thread
    mainThread: Thread
    idleThread: Thread
    threadsToBeDestroyed:  List [Thread]
    currentInterruptStatus: int
    processManager: ProcessManager
    threadManager: ThreadManager
    frameManager: FrameManager
    --diskDriver: DiskDriver
    --serialDriver: SerialDriver
    --fileManager: FileManager

  functions

    -- These routines are called from the Runtime.s assembly code when
    -- the corresponding interrupt/syscall occurs:

    TimerInterruptHandler ()
```

```
    DiskInterruptHandler ()
    SerialInterruptHandler ()
    IllegalInstructionHandler ()
    ArithmeticExceptionHandler ()
    AddressExceptionHandler ()
    PageInvalidExceptionHandler ()
    PageReadonlyExceptionHandler ()
    PrivilegedInstructionHandler ()
    AlignmentExceptionHandler ()
    SyscallTrapHandler (syscallCodeNum, arg1, arg2, arg3, arg4: int) returns int

    -- These routines are invoked when a kernel call is made:

    Handle_Sys_Fork () returns int
    Handle_Sys_Yield ()
    Handle_Sys_Exec (filename: ptr to array of char) returns int
    Handle_Sys_Join (processID: int) returns int
    Handle_Sys_Exit (returnStatus: int)
    Handle_Sys_Create (filename: String) returns int
    Handle_Sys_Open (filename: String) returns int
    Handle_Sys_Read (fileDesc: int, buffer: ptr to char, sizeInBytes: int) returns
int
    Handle_Sys_Write (fileDesc: int, buffer: ptr to char, sizeInBytes: int) returns
int
    Handle_Sys_Seek (fileDesc: int, newCurrentPos: int) returns int
    Handle_Sys_Close (fileDesc: int)
    Handle_Sys_Shutdown ()
InitializeScheduler ()
    Run (nextThread: ptr to Thread)
    PrintReadyList ()
    ThreadStartMain ()
    ThreadFinish ()
    FatalError_ThreadVersion (errorMessage: ptr to array of char)
    SetInterruptsTo (newStatus: int) returns int
    ProcessFinish (exitStatus: int)

    -- Routines from Switch.s:

    external Switch (prevThread, nextThread: ptr to Thread)
    external ThreadStartUp ()
    external GetOldUserPCFromSystemStack () returns int
    external LoadPageTableRegs (ptbr, ptlr: int)  -- Execute "LDPTBR" and "LDPTLR"
    external SaveUserRegs (p: ptr to int)          -- Execute "readu" instructions
    external RestoreUserRegs (p: ptr to int)       -- Execute "writeu" instructions

    -- The following routine sets the "InterruptsEnabled" bit, sets the
    -- "PagingEnabled" bit, clears the "SystemMode" bit, and jumps to the
    -- address given by "initPC".
    external BecomeUserThread (initStack, initPC, initSystemStack: int)


    -------------- Semaphore ---------------

  class Semaphore
    superclass Object
    fields
      count: int
      waitingThreads: List [Thread]
    methods
      Init (initialCount: int)
      Down ()
      Up ()
  endClass

    -------------- Mutex ---------------

  class Mutex
    superclass Object
    fields
      heldBy: ptr to Thread              -- Null means this mutex is unlocked.
      waitingThreads: List [Thread]
    methods
      Init ()
      Lock ()
      Unlock ()
      IsHeldByCurrentThread () returns bool
endClass
```

```
          -------------- Condition --------------

      class Condition
        superclass Object
        fields
          waitingThreads: List [Thread]
        methods
          Init ()
          Wait (mutex: ptr to Mutex)
          Signal (mutex: ptr to Mutex)
          Broadcast (mutex: ptr to Mutex)
      endClass

      -------------- HoareCondition --------------

      class HoareCondition
        superclass Object
        fields
          waitingThreads: List [Thread]
        methods
          Init ()
          Wait (mutex: ptr to Mutex)
          Signal (mutex: ptr to Mutex)

      endClass

          -------------- Thread --------------

      class Thread
        superclass Listable
        fields
          -- The first two fields are at fixed offsets, hardwired into Switch!
          regs: array [13] of int        -- Space for r2..r14
          stackTop: ptr to void          -- Space for r15 (system stack top ptr)
          name: ptr to array of char
          status: int                    -- JUST_CREATED, READY, RUNNING, BLOCKED,
UNUSED
          initialFunction: ptr to function (int)   -- The thread's "main" function
          initialArgument: int                     -- The argument to that function
          systemStack: array [SYSTEM_STACK_SIZE] of int
        isUserThread: bool
          userRegs: array [15] of int    -- Space for r1..r15
          myProcess: ptr to ProcessControlBlock
        methods
          Init (n: ptr to array of char)
          Fork (fun: ptr to function (int), arg: int)
          Yield ()
          Sleep ()
          CheckOverflow ()
          Print ()
      endClass

      -------------------------- ThreadManager -------------------------------
      --
      --  There is only one instance of this class, created at startup time.
      --
      class ThreadManager
        superclass Object
        fields
          threadTable: array [MAX_NUMBER_OF_PROCESSES] of Thread
          freeList: List [Thread]
          threadManagerLock:Mutex
          aThreadBecameFree:Condition
        methods
          Init ()
          Print ()
          GetANewThread () returns ptr to Thread
          FreeThread (th: ptr to Thread)
      endClass

      -------------------------- ProcessControlBlock ----------------------------
----
      --
      --  There are a fixed, preset number of these objects, which are created at
      --  startup and are kept in the array "ProcessManager.processTable".  When
      --  a process is started, a ProcessControlBlock is allocated from this
```

```
--  array and the state of the process is kept in this object.
--
class ProcessControlBlock
  superclass Listable
  fields
    pid: int                      -- The process ID
    parentsPid: int               -- The pid of the parent of this process
    status: int                   -- ACTIVE, ZOMBIE, or FREE
    myThread: ptr to Thread       -- Each process has one thread
    exitStatus: int               -- The value passed to Sys_Exit
  addrSpace: AddrSpace            -- The logical address space
    -- fileDescriptor: array [MAX_FILES_PER_PROCESS] of ptr to OpenFile
  methods
    Init ()
    Print ()
    PrintShort ()
endClass

-------------------------- ProcessManager --------------------------------
--
--  There is only one instance of this class, created at startup time.
--
class ProcessManager
  superclass Object
  fields
    processTable: array [MAX_NUMBER_OF_PROCESSES] of ProcessControlBlock
    processManagerLock: Mutex               -- These synchronization objects
    aProcessBecameFree: Condition           --    apply to the "freeList"
    freeList: List [ProcessControlBlock]
    aProcessDied: Condition                 -- Signalled for new ZOMBIEs
    nextPid: int
  methods
    Init ()
    Print ()
    PrintShort ()
    GetANewProcess () returns ptr to ProcessControlBlock
    FreeProcess (p: ptr to ProcessControlBlock)
    --TurnIntoZombie (p: ptr to ProcessControlBlock)
    --WaitForZombie (proc: ptr to ProcessControlBlock) returns int
endClass

-------------------------- FrameManager --------------------------------
--
--  There is only one instance of this class.
--
class FrameManager
  superclass Object
  fields
    framesInUse: BitMap
    numberFreeFrames: int
    frameManagerLock: Mutex
    newFramesAvailable: Condition
    waitThread: Condition

  methods
    Init ()
    Print ()
    GetAFrame () returns int                        -- returns addr of frame
    GetNewFrames (aPageTable: ptr to AddrSpace, numFramesNeeded: int)
    ReturnAllFrames (aPageTable: ptr to AddrSpace)
endClass

-------------------------- AddrSpace --------------------------------
--
--  There is one instance for every virtual address space.
--
class AddrSpace
  superclass Object
  fields
    numberOfPages: int
    pageTable: array [MAX_PAGES_PER_VIRT_SPACE] of int
  methods
    Init ()
    Print ()
    ExtractFrameAddr (entry: int) returns int
    ExtractUndefinedBits (entry: int) returns int
    SetFrameAddr (entry: int, frameAddr: int)
```

```
            IsDirty (entry: int) returns bool
            IsReferenced (entry: int) returns bool
            IsWritable (entry: int) returns bool
            IsValid (entry: int) returns bool
            SetDirty (entry: int)
            SetReferenced (entry: int)
            SetWritable (entry: int)
            SetValid (entry: int)
            ClearDirty (entry: int)
            ClearReferenced (entry: int)
            ClearWritable (entry: int)
            ClearValid (entry: int)
            SetToThisPageTable ()
            CopyBytesFromVirtual (kernelAddr, virtAddr, numBytes: int) returns int
            CopyBytesToVirtual (virtAddr, kernelAddr, numBytes: int) returns int
            GetStringFromVirtual (kernelAddr: String, virtAddr, maxSize: int) returns int
    endClass

endHeader
```