Mrudhula A. Shenava SUID: 265469976

PRINCIPLES OF OPERATING SYSTEMS

ANSWERS

1) What are the possible process states in JNachos and where are they defined (file:linenumber)?
   - NachosProcess.java:(20-22)

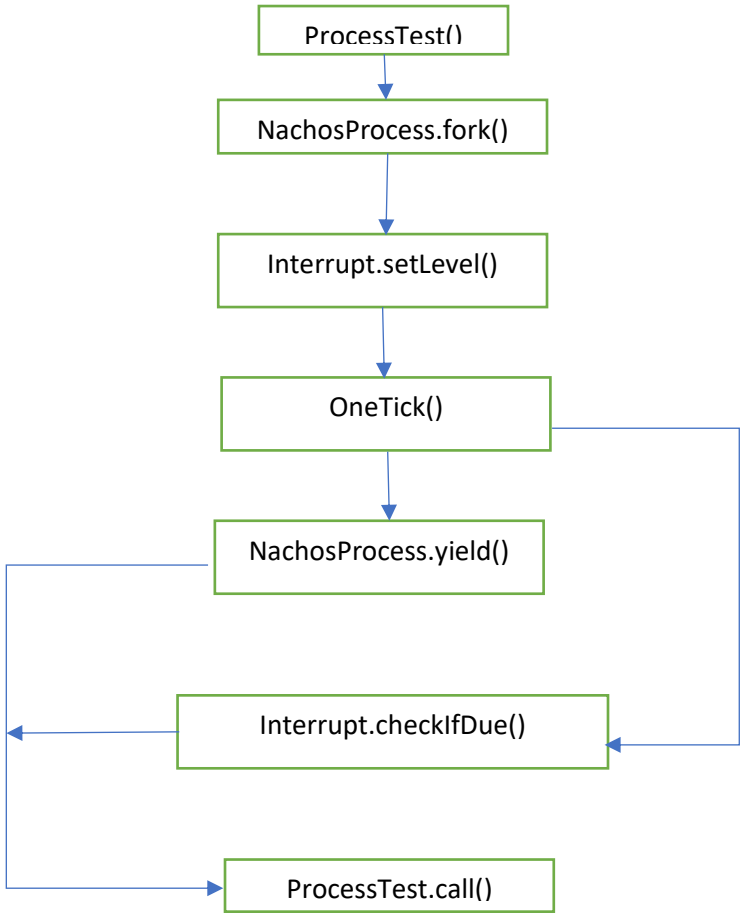
2) Where do the transitions between these states occur? (file:linenumber)
   - NachosProcess.java:(345-390)

3) Describe which type of policy the Scheduler is using to schedule NachosProcesses. (We do not need a specific name of a scheduling algorithm, just a natural language description of how it chooses the next thread to run).
   - FIFO

4) Open ProcessTest.java in the kern directory. How many lines are printed when ProcessTest is run?
   - 25

5) In ProcessTest.java, how many lines are printed when we change:
   for(int num = 0; num < 5; num++)
   to
   for(int num = 0; num < which; num++)
   Why is that?
   - 10

6) Draw the call graph from ProcessTest() to the function ”call” in process test. (This is done by looking through the source code to see how functions call each other).



1) What is the purpose of the Machine::Run function?
   - The Machine::Run function executes the decoded instructions one by one. It also mentions at what time each process has been launched. While executing, the status is set to ‘UserMode’. Then, each user code is executed.

2) What does the emulated main memory look like (datatype)in JNachos?
   - String data type.

3) How much RAM does JNachos machine have?
   - JNachos has a RAM of 128 KB.

4) How can I run a user program in JNachos? (At the command line).
   - To run a user program in JNachos at command line we use ‘-x <nachos file>’

5) Why is the last line in JNachos::startProcess never executed?
   - It states that the 'Machin.run()' never returns any value. The address space exits by doing the system call exit.

6) Which class maintains the memory map and loads executable memory images from the file system?
   - NachosFileSytaem class maintains the memory map and loads executable memory images from the file system.

7) What system calls are currently implemented in JNachos?
   - System calls that are currently implemented in JNachos are:
     Halt
     Exit

8) Is JNachos currently multiprogrammed or uniprogrammed?
   - JNachos is currently uniprogrammed.

9) What is the purpose of the Timer object in JNachos?
   - The Timer object in JNachos generates a CPU interrupt every X milliseconds. It is used for time-slicing.

10) Explain the call graph from executing user instructions to running a system call implementation.
   - The executable file in the user program is opened.
   - If the file doesn't exist, then an error is thrown.
   - Else, if the file is stored in memory storage and the address is allocated to that file.
   - Each instruction is executed one by one.
   - The instructions are executed individually and if any interrupt or exception occurs,the exception handler is called.
   - Once the exception is handled, the instruction is executed again, from the beginning, in case the status of the file has changed.

11) Why can interrupts occur at every user-instruction versus only when interrupts go from off to on in Kernel mode?
   - Interrupts occur at every user-instruction as the UserMode runs on the stimulated hardware and has the privilege to provide user interrupts. Whereas, the Kernel mode runs on Operating system and executes CPU instructions which are binary digits. Hence, the interrupts go off to on.

12) If the user space/mode runs on the simulated hardware, where does kernel space/mode run?
   - The user space/mode runs on the simulated hardware, where does kernel space/mode run on the Operating system

13) If we consider JNachos to be a process running on top of the JVM on an arbitrary operating system, then what are NachosProcesses? If we consider JNachos to be an Operating system, what are NachosProcesses?
    - If JNachos is to be running on top of the JVM on an arbitrary operating system, then NachosProcesses are threads. But if we consider JNachos to be an Operating system, the NachosProcesses will be a Process.

14) Explain why we do not see the actual changing of registers in JNachos when switching between kernel processes.
    - To modify the registers we need assembly language functions. Hence, we do not see the actual changing of registers in JNachos when switching between kernel processes as it is written in Java.