**CSI 4500 Operating System**

**Homework1**

**[total: 50 points]**

**Name: \_\_\_Fabian LeFevre\_\_\_**

Question 1: **Briefly** explaining **WHAT CONDITIONS** cause a process to move between each of the following 3 states indicated by each arrow (from 1 to 6). Label it N/A if it doesn’t happen. [12 points: each correct one will earn 2 points]

1

2

3

4

5

6

1. Interrupt
2. Scheduler Dispatch
3. N/A
4. I/O
5. N/A
6. Event wait

Question 2. In this question, events are given during the execution of a *grading* program. You are supposed to understand the process state transition and fill out those blanks and choose the right options. Hint: state transition occurs when some particular events happen. Please use one of *ready*, *running*, *waiting* states as the possible state for the process. When you need to determine the running mode, please use either *user* or *kernel*.[Please check and understand these concepts through reading either from the slides or textbook] [19 points: one for each cell in the answer table]

  intexam\_grade (int m1, int m2, int m3)

 {

intexam\_sum;

exam\_sum = 0.3\*m1 + 0.5\*m2 + 0.2\*m3;

returnexam\_sum;

  }

-- **Event**: user runs the program, the process is in **Q1**\_**running**\_ state.

  main ()

  {

-- **Event**: when scheduler chooses this program to run**Q2**, the process is in \_\_**ready**\_\_state and in user kernel mode. (**Q3**, please choose the right mode from the above options)

    for (i=1; i<=24; i++)

    { scanf ("%d %d %d", &t1, &t2, &t3);

-- **Event**: I/O statement **Q4**, the process is in \_\_**waiting**\_\_ state.

Before state transition, switch to user kernel mode to handle scanf. (**Q5**, please choose the correct mode).

**Q6**, in the multiprogramming environment, will the OS switch to execute another process? Yes [ **YES** ] No [ ]

-- **Event**: I/O is done, I/O device sends an interrupt. Interrupt makes the current running process stops temporarily.**Q7**,CPU switches to interrupt handler and puts this process to \_\_\_**ready**\_\_\_ state. CPU continues to execute the original running process (we are assuming non-preemptive scheduling is used)

-- **Event**: scheduled to run This process runs. (There is a context switch)

      scanf ("%d %d %d ", &P1, &P2, &P3);

      scanf ("%d %d %d", &m1, &m2, &m3);

      scanf ("%d %d %d %d", &h1, &h2, &h3, &h4);

-- **Event**: I/O statement **Q8**, the process is in \_\_\_**waiting**\_\_\_ state.

-- **Event**: I/O completion **Q9**, the process is in \_\_\_\_**ready**\_\_\_\_ state.

-- **Event**: scheduled to run  running (context switch)

      Project\_total = (P1+P2+P3);

      Homework\_total = (h1+h2+h3+h4) / 4;

      Exam\_total = exam\_grade(m1, m2, m3);

-- **Event**: procedure call, the process is in \_\_\_**running**\_\_\_ state and

the program is in user kernel mode. (**Q10**, please answer the state of the process, and choose the right mode from the above options)

      total1 = (t1+t2+t3)\*0.1+Project\_total\*0.50 + Homework\_total\*0.1 + Exam\_total\*0.3);

-- **Event**: interrupted due to time quantum expiration **Q11**, the state of the process will be changed from \_\_\_**running**\_\_\_state to \_\_\_**ready**\_\_\_state, before the transition, the CPU will handle the timeout and switch to execute another process (context switch)

-- **Event**: scheduled to run **Q12**the state of process will be changed from \_\_\_**ready**\_\_\_state to \_\_**running**\_\_state(context switch)

      total2 = scale\_the\_grade(Project\_total, Homework\_total, m1, m2, m3);

      if (total1 > total2) total = total1;

      else total = total2;

-- **Event**: I/O device interrupts to signal the completion of an I/O of a different process

         Switch to user kernel mode to run interrupt handler (context switch) (**Q13**, please check the correct one from the above options). **Q14**, the current process is in \_\_**ready**\_\_ state.

          Switch back to run this process in user kernel mode(context switch)**Q15**

      if (extra = 0) final = total;

      else final = (float) Bonus(total, extra);

        printf (" %.1f ", final);

        scanf ("%c", &ch); printf ("\n");

    }

    fork ();

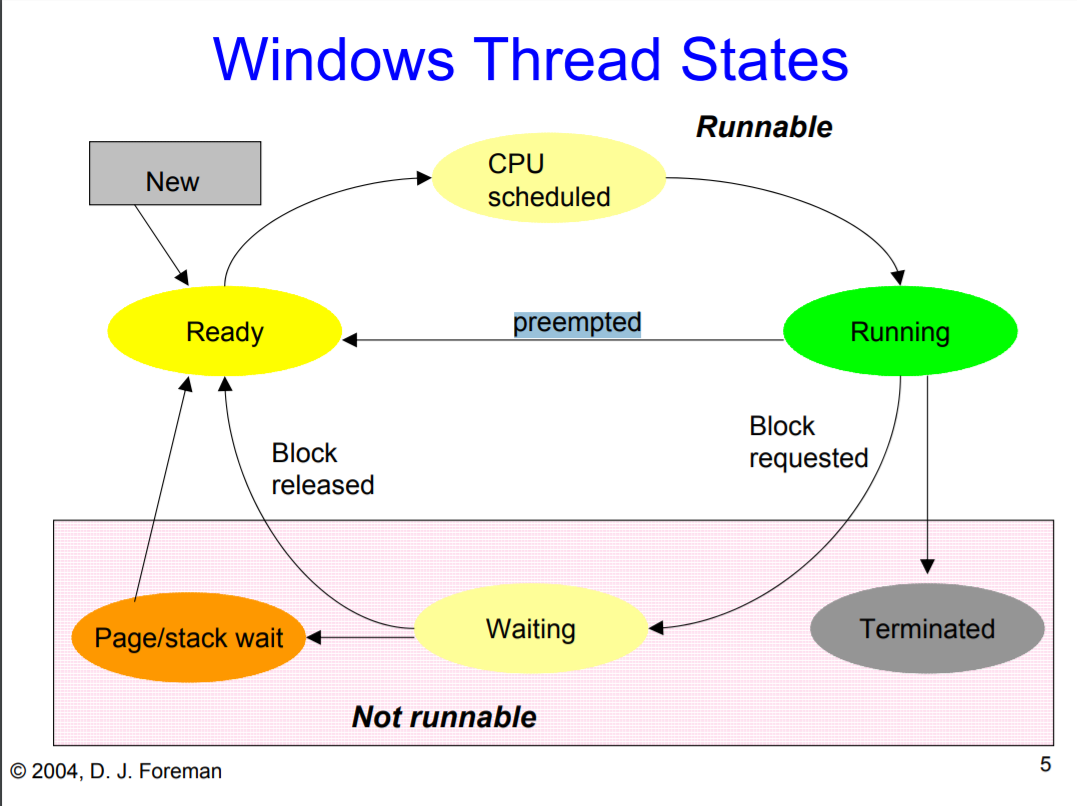
-- **Event**: system call

          Kernel trap, switch to kernel mode to handle fork (context switch)

**Q16**, OS creates the child process, put it in \_\_**ready**\_\_ queue

}

Question 3. Case study, in this task, you need to study a process state transition for any real system (linux, wondows, mac, android, ios, etc.). What you need to do is draw (or copy paste) the process state transition diagram and describe the conditions for every transition.In case of “copy and paste”, please note the link [19 points]



<http://dforeman.cs.binghamton.edu/~foreman/OS-common/DJ-slides/W14/CS350-state-diags.pdf>

**(SLIDE 5/6)**

**New to Ready:** Process created and finished initializing.

**Ready to CPU Scheduled:** Process acknowledged and set for scheduling.

**CPU Scheduled to Running:** CPU scheduler passes process to CPU resources and begins running the process’ instructions.

**Running to Ready:** Process interrupted, forced to stop, and releases the CPU resources, but is only waiting to be rescheduled by the CPU.

**Running to Waiting:** Process stopped, requests for a block, but is waiting for another process, I/O action, or some other event to be completed.

**Running to Terminated:** Process finishes all instructions, releases CPU resources, and moves to terminate itself.

**Waiting to Ready:** The action or event completes, process released from the block, and moves to ready for the CPU scheduler to reschedule it.

**Waiting to Page/Stack Wait:** Process is waiting for portions of the stack or page to finish before moving to ready.

**Page/Stack Wait to Ready:** The actions or instructions needed in the stack or page to finish are completed and thus the process moves to ready so that the CPU Scheduler can reschedule it.