Exam 1, CS – 405, Fall 2020
Iyengar

Your Name: ___**Adam Stammer**_________________

Please be neat. You may use both sides of the exam sheets.

List the different responsibilities of an OS? (10)
Manage Hardware Resources
Protect Hardware
Run/Schedule Jobs
Control IO
File Management

Describe the single-user system of the early days of computing. How was this modified to implement Automatic Job Sequencing, and then Batch Processing? (15)
Originally, jobs were loaded one at a time and had to frequently wait for more IO of some kind. These single jobs also had to be handled by a user once they were done. This left the CPU idling for extended periods of time which is obviously not an efficient way to do things. Since computers were quite expensive at the time, users were even more motivated to fix this inefficiency. Automatic Job Sequencing allowed these systems to work through a list of jobs one by one so as to avoid the idle time associated with waiting for the user to come back, especially since users didn't always know how long their job would take. Batch Processing allowed the computer to actively have multiple jobs in memory and then multiplex through the entire batch of jobs. This drastically helped avoid idle time associated with waiting for resources/IO.

OS is an interrupt driven program. Explain. (5)
Interrupts are a generally hardware driven feature that allows for something to be done "right now", without waiting for the software to finish its current task. When hardware based, it also means the CPU doesn't have to actively check or listen for the interrupt to occur which saves a lot of CPU time. When something triggers the interrupt, the machine goes through a predetermined process of events, frequently instructions. In the context of an OS, this is often used to switch from one job to the next. The hardware interrupt triggers a context switch to another job.

OS is implemented using a set of policies and mechanisms. Explain the difference between mechanisms and policy. Why the separation? (10)
Mechanisms tell us how to do something. These are often hardware and software based mechanisms that are not meant to be changed, if they even can be, by any users. Spending a dedicated amount of time on each job is an example of a mechanism. On the other hand, policies tell us what is to be done, usually in a specific and quantifiable way. Policies are much more likely to change from on place to another, and with time. How much time to spend on each job is an example of a policy decision.

What is meant by multiprogramming? (5)
Multiprogramming is a method of increasing computer efficiency by running multiple jobs at the "same time". They don't necessarily run at the exact same time, but rather the processor takes turns running a some of one program, then some of the next, and so on. How this is accomplished is a major aspect of an OS. This relies on some form of scheduler to determine which jobs to run, and for how long to run them. Tweaking this system can result in significant and very tangibly different results for the user.

What is meant by a Time-sharing system? What is its intended/expected effect on the user? (10)
Time-sharing systems switch between multiple jobs very frequently such that it appears that all of the jobs are running simultaneously. The intended goals of such a system is that the user(s) don't have to wait for a program to respond. This system also functions to increase efficiency of the CPU since it can work on other processes while waiting for input on another, rather than sitting idle waiting.

Why do we have instructions for interrupt enable and interrupt disable? (10)
Some interrupts are not always important enough to deal with all the time. If code is running to recover from an error, like correcting a memory error, you don't want just any interrupt to take over the CPU. You want to fix the error first, and then allow interrupts to do as they need to. Some interrupts are too important to risk an interrupt occurring at the same time. The ability to disable and enable interrupts is sometimes required to maintain the stability of these interrupts and thus the system as a whole.

What are system calls? What is the advantage of using system calls over interrupts? Give examples of system calls? (15)
System Calls are an example of software interrupts. They allow a user program to request services, actions, or resources from the OS itself, that the user program generally doesn't have access to. IO is a common example of this, as user programs usually don't have direct access to talk to a monitor, or a disk, etc. The program must use system calls to request that the OS interact with the IO devices and pass IO between the user program and those IO devices.

How is system boot initiated and accomplished? (5)
Booting is often handled by a bootstrapper. This is a program that locates the OS and loads it into main memory. Since this process can be harmful to the system if tampered with or done wrong, this bootstrapper is often stored in some for of ROM not normally accessible by the rest of the system. Once it successfully loads the OS into memory and starts it, the bootstrapper has completed its job and the OS takes over running through its own startup routines.

What is meant by buffering? What hardware feature is essential for buffering? (10)

Buffering is the process by which a system can load data/io for a process during the execution of that process. This requires additional hardware hardware in the for of DMA (Direct Memory Access) since the memory can't normally be written to while the cpu is reading form it. DMA gives the system an additional location in which to load data during execution, that can then be transferred into memory much much faster, once the CPU is done executing its current task.

What is meant by Spooling? (5)
Spooling is the process by which a system can load data/io for a process while executing a different process. Preemptively loading the next job can avoid large amounts of idle time as the CPU can start executing the next job much faster than if it had to wait to load it into memory first.

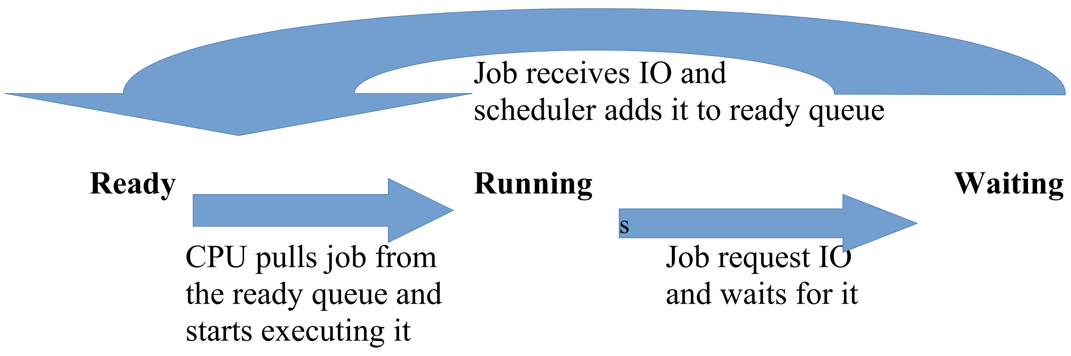
Argue for the need for dual-mode operation. Describe how this is implemented. List three (3) privileged operation. (15)

Multiple User modes help protect the hardware by keeping user programs from directly accessing it. Privileged instructions can only be used by the kernel itself, and a user program must request they be done by the kernel if the user program wants to use them. If a user program attempts to use these privileged instructions directly, they will be considered illegal instructions and they won't execute.

Examples of Privileged Instructions:
IO Handling
Interrupt Handling
User Mode Switching

List the essential information stored in a process control block (at least 10).
Process State
Process Identifier
Process Privileges
Program Counter
CPU Registers
Scheduling Information
Memory Management Information
Accounting Information
IO Status
List of open files

Show the basic state transition diagram for a process. (10)


Job receives IO and scheduler adds it to ready queue
Ready
Running
Waiting
CPU pulls job from the ready queue and starts executing it
Job request IO and waits for it

Describe the differences in the actions taken by a short-term scheduler and a long-term scheduler. (10)
The long term scheduler is generally more interested in entire Jobs, and deciding which ones to load into memory (which job should we start next). The short term scheduler, on the other hand, is more focused on which jobs the cpu should run next, based on the ready queue (which job is currently ready to run). As such the LTS is more concerned with what is and isn't in memory, while the STS is focused more on the CPU and the states of the jobs that are already in memory, with no concern for what has yet to be loaded into memory. As the name implies, this also means that the STS executes much more frequently than the long term scheduler.

List some of the process events that invoke the short-term scheduler. (5)
The state change of a process
CPU being idle
New process being loaded into memory
When a job finishes

What is meant by pre-emption? Give an example when a process is pre-empted. (5)
Pre-emption is when a job is scheduled to run before the cpu is actively waiting for a task to run. If a job was waiting for IO, received that IO, and then switched into the ready state, that job could be premptively scheduled for execution once the cpu is freed up.

Assume the following jobs are in the ready-queue with the associated cpu burst.

P1 / 120, P2 / 60

Execute the jobs using the following algorithm and timer values:

FCFS (Round Robin) with timer value of 20

Calculate the completion time for each of the job, average wait time for all the job, and the total number of context switching that has to be done to complete all the jobs. (20)

	p1	**p2**	
0	**120**	**60**	
20	**100**	**60**	**cw**
40	**100**	**40**	**cw**
60	**80**	**40**	**cw**
80	**80**	**20**	**cw**
100	**60**	**20**	**cw**
120	**60**	**0 P2 done**	**cw**
140	**40**		
160	**20**		
180	**0 P1 done**		

P1 Done at time 180
P2 Done at time 120
Average Time of Execution: 150
Number of Context Switches: 6
Average Wait Time For All Jobs:
60