Kamesh Vedula 68149804

CS 143A: Principles of Operating Systems

Homework #1

Due Date: Fri, 20 April 2018, 11:55 PM

Please submit a *PDF file* containing all answers to EEE+ Canvas. Total Marks=100

Question 1: Polling, Interrupts, Traps [6, 2, 2, 6, 4]

a. [6 pts]) Explain the relationship and/or differences between the following concept pairs:

1. Synchronous vs. asynchronous I/O -

Synchronous I/O : wait instruction idles CPU until I/O is complete, no simultaneous I/O processing.

Asynchronous I/O : after I/O is initiated, control returns to user program w/o waiting for I/O completion.

2. Interrupts vs. polling -

Polling : requires the CPU to repeatedly read the busy bit in the status register of a device controller.

Interrupt : The interrupt request line is a wire in CPU hardware, checked after every instruction.

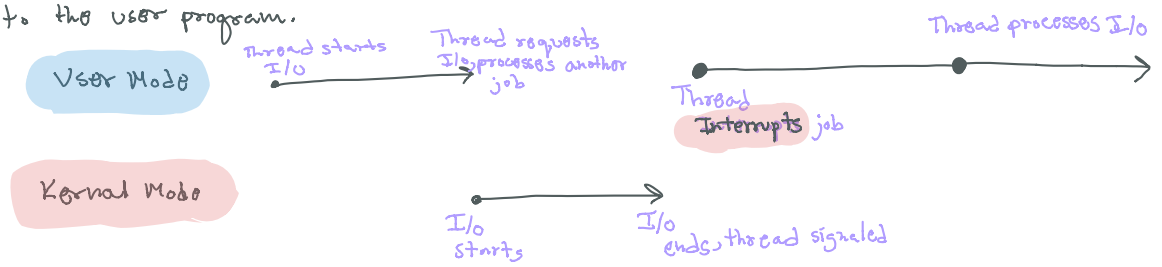
3. Traps vs. interrupts -

A trap is a software-generated interrupt caused by an error or a user request.

b. [2 pts]) Explain which of the following I/O methods is most likely interrupt driven. Why?

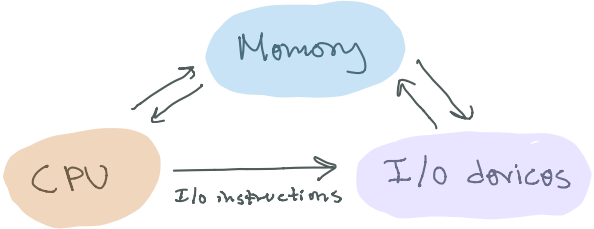
- Synchronous I/O
- Asynchronous I/O

Asynchronous I/O, since it has the ability to not have to complete the I/O for it to return control to the user program.



c. [2 pts]) Briefly explain (1-2 sentences) how DMA improves I/O efficiency.

The DMA is used for high speed I/O devices able to transmit information at close to memory speeds. The device controller transfers blocks of data from buffer storage directly to main memory w/o CPU intervention. Only one interrupt ends up being generated per block as opposed to one per byte.



d. [6 pts]) A laser printer produces up to 30 pages per minute, where a page consists of 5000 characters. The system uses interrupt-driven I/O by raising an interrupt for every printed character. For the following values of how long each interrupt takes to process (in microseconds), solve for and answer the following questions (show your work for full credit!):

- How much CPU time will be spent processing interrupts **(in %)**?
- If a polling-driven implementation takes up 10% of the CPU time to process this I/O, would it be more efficient to use polling instead of interrupts? Explain why or why not.

	Interrupt proc. time	**% of CPU time spent processing interrupts**	**Polling or interrupts? Why?**
1)	50 microsecs.	7.5/60=0.125 0.125*100 = 12.5%	Polling-driven implementation would be better since it takes 10% of time vs. the 12.5% that interrupt-driven implementation would take.
2)	20 microsecs.	3/60 = 0.05 0.05 · 100 = 5%	Interrupt-driven implementation would be better b/c it take 5% of CPU time vs. the 10% polling-driven implementation would take in this case.

e. [4 pts]) Assume a mouse generates an interrupt whenever its position changes by 0.2 mm. It reports each such change by placing 4 bytes of data into a buffer accessible by the CPU. At what rate **(in KB/sec)** must the CPU retrieve the data if the mouse moves at a speed of 10 cm/second in order to read all the data ?

10/0.02 = 500 => 500 * 4 = 2000 ≈ 2kb/s

Question 2: Parallel, Distributed, Real-time Systems [4, 6, 10]

a. [4 pts]) Briefly explain the difference between multiprogramming and multiprocessing.

Multiprogramming => Multiple Jobs or Processes

Multiprocessing => Multiple CPUs

b. [6 pts]) Give two real-world examples for distributed computing system applications and explain why the systems are distributed.

1. Google.com : google uses a server farm to execute its processing. There are sets of servers that fetch, 'read', take your input and display the web pages.

2. Facebook.com : again has large sets of server farms to process all the data that each user has.

c. [10 pts]) For each of the following systems, indicate whether they are a) hard real-time systems, b) soft real-time systems, or c) not real-time systems at all by checking ☑ the appropriate boxes. Give one sentence for each of your choices to briefly explain why.

		Hard real-time	Soft real-time	Not real-time	Brief explanation
1.	Skype		✓		It's ok if the video may be blurry or not loading and can be fixed by getting better connection.
2.	YouTube		✓		It's not the end of the world if a video doesn't load or work.
3.	Adaptive cruise control for autonomous vehicles	✓			This hardware & software has to operate within a stringent deadline. It is considered to have failed if it does not complete it within the time slot. Failure can cause accidents.
4.	High-frequency stock trading system	✓			If it fails someone loses a lot of money and that would be very bad.
5.	Uploading your Homework PDF to Canvas			✓	It's ok if it does not work sometimes and the file need not be ready to view immediately after submission. Failure is ok.

Question 3: System Calls [7.5, 7.5, 5]

a. [7.5 pts]) For each of the brief description about the following system calls used in Unix, write the name of the system call and its parameters. (Hint: Use the *man* pages.)

a. Create a child process

fork() ; no parameters

b. Wait for process termination

wait(status) ; output data from child to parent

c. Get process ID

getpid(); no parameters

b. [7.5 pts]) Suppose you want to write a program that opens the file passed to it on the command line (argv[1]) and writes another file (argv[2]) back out (with the same contents as the first file). List 3 system calls that will be used and explain their purpose in this program.

open : open the first file which is argv[1]

read : read the contents of the opened file

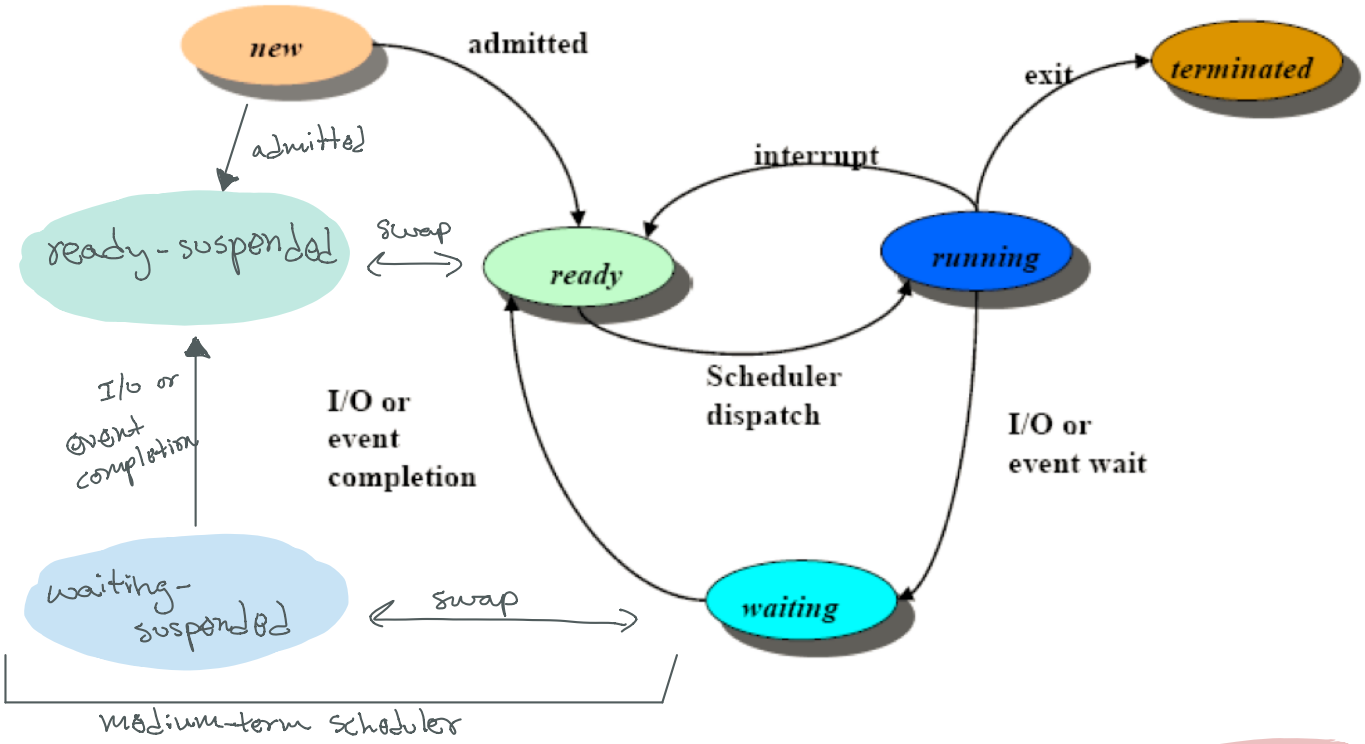
write: write what was read onto another file (argv[2]) and return it.

c. [5 pts]) For each of the following operations, indicate whether they should be privileged or not by checking ☑ the appropriate boxes.

	Yes	No
Searching a string for a particular sequence		✓
Generate an interrupt	✓	
Process hardware interrupt	✓	
Check list of currently running processes		✓
Access network card buffer		✓

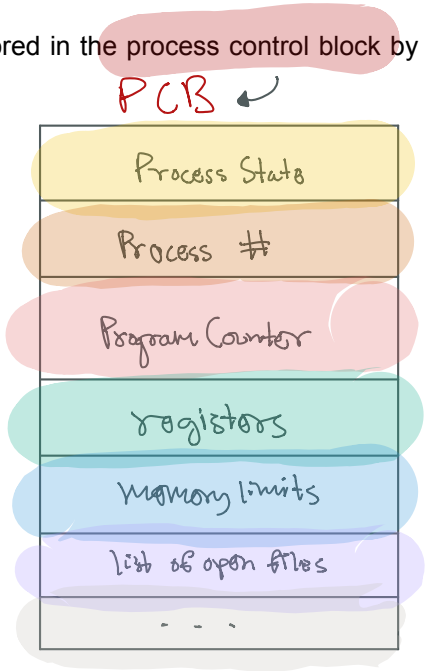
Question 4: Processes [15, 5]

a. [15 pts]) The following figure shows all states and state transitions during the lifetime of a process (see figure below). As discussed in class, a medium-term scheduler can swap a process in or out of memory. When a process is swapped out, it is *suspended*.
Modify the state diagram below by drawing in all new states and state transitions (using directed edges) that are necessary to accommodate the new *suspended* state. Also name the conditions under which the state transitions you added should occur.



b. [5 pts]) For each of the following, indicate whether they are stored in the process control block by checking ☑ the appropriate boxes.

	Yes	No
CPU registers	✓	
Number of processes		✓
Program counter	✓	
Process ID	✓	
File permissions		✓



Question 5: Threads[3, 4, 9, 4]

a. [3 pts]) Why are threads generally faster to create than processes?

There are many "tables" that a process must have and if you have a process, all these tables have to be initialized. If a thread is created, you don't have to initilize all the tables since it will use the ones of its processes.

b. [4 pts]) Suggest one application that might benefit from a multi-threaded implementation and briefly explain why.

A web Browser, such as Safari because you have to dispatch multiple to load multiple HTML references in parallel during a single page load. This allows us to make the webpage load faster and maximize the TCP data throughput.

c. [9 pts]) Briefly discuss the switching costs for the following scenarios:

a. Switching between two *user-level threads* in the same process.

Cheap and fast, threads do not have to call OS and cause interrupts to kernal.

b. Switching between two *user-level threads* in different processes.

Should be cheap and pretty fast as well since there is no need to talk to the kernal-level threads.

c. Switching between a *user-level thread* and a *kernel-level thread.*

without kernal involvement, switching is fast, but with a kernal-level thread it is slower and more expensive.

d. [4 pts]) When a process is multithreaded, indicate which of these resources are shared among the threads, and which are private to each thread by checking ☑ the appropriate boxes:

	Shared	Private
Stack memory		✓
Open files	✓	
CPU registers		✓
Global variables	✓	