**CSE3502, Operating Systems**

**Fall 2019, Homework 1**

**1. What are the three themes of an operating system? Explain each of the function briefly. (5 point)**

*Three themes of operating systems: virtualization, concurrency and persistence.*

1. *A software layer between the hardware and the application programs/users which provides a virtualization interface.*
2. *A resource manager that allows concurrent programs/users to share the hardware resources.*
3. *And it ensures information stored persistently in the computer.*

**2. What are the differences of orphan processes and zombie processes? (5 point)**

*When a parent process dies, child process becomes an orphan process.*

*When a child dies, a SIGCHLD signal is sent to the parent. If parent doesn’t wait() on the child, and child exit()s, it becomes a zombie.*

**3.** **What are the differences between a monolithic kernel and a microkernel? (5 point)**

*Monolithic kernels*

*All functionalities are compiled together*

*All code runs in privileged kernel-space*

*Microkernels*

*Only essential functionalities are compiled into the kernel*

*All other functionalities run in unprivileged user space*

**4. What is system call used for? (5 point)**

*A system call is a way for programs to interact with the operating system*

*A computer program makes a system call when it makes a request to the operating system's kernel*

*System call provides the services of the operating system to the user programs via Application Program Interface (API)*

**5. What is a process? What are the two essential parts of a process? How is a process different from a program? (5 point)**

*A process is an abstraction of a running program. It has two essential parts: the sequential execution of instructions and the process state.*

*A program is represented by static code and data, while a process is a dynamic instantiation of a program.*

**6. Given the five-state process model, explain how does a process transit among these states and on what events? (5 point)**

*Process creation: 🡪 new*

*Process admission: new 🡪 ready*

*Scheduler dispatch: ready🡪 running*

*Timer interrupt or forced preemption: running 🡪 ready*

*I/O or event wait: running 🡪 blocked*

*I/O or event completion: blocked 🡪 ready*

*Exit or killed: running🡪 terminated*

**7. What are the differences of threads and processes? (5 point)**

*Processes have their own address spaces. Threads share address spaces including shared code, data, heap, and files. Threads are inexpensive to create and to do context switch. However, processes usually have better isolation and protection from other processes.*

**8. Discuss the advantages and disadvantages of user-level threads and kernel-level threads. (15 point)**

*User-level threads*

*Advantages:*

*1. No OS threading-supporting is needed*

*2. Lightweight, thread switching is inexpensive, no trap to kernel*

*3. Each process has its own customized scheduling algorithm*

*Disadvantages:*

*1. Hard to implement blocking call in threads*

*2. Hard to deal with page faults*

*3. No clock interrupt, hard to implement preemption*

*Kernel-level threads*

*Advantages:*

*1. Known to OS scheduler*

*2. Blocking calls and page faults do not block the whole process*

*Disadvantages:*

*1. Slow due to the trap to the kernel*

*2. Expensive to create and switch*

**9. What is the difference between interrupt and polling? What are the possible issues? What is the deadlock? (10 point)**

*Polling: repeatedly check whether the hardware is ready or not*

*Interrupt: hardware signals to the kernel when attention is needed*

*When two or more threads stop making progress indefinitely because they are all waiting for each other to do something.*

**10. What are the commonalities and differences between semaphore and mutex? (5 point)**

*A semaphore is a variable that counts the number of pending wakeups. If properly configured, semaphores allow a predefined number of threads to enter a critical region. Mutex is a simplified version of semaphore (binary semaphore). It is only useful in cases of mutual exclusion. Both semaphore and mutex provide a sleep-and-wake locking approach to mutual exclusion.*

**11. List different ways to avoid race conditions. (5 point)**

*Disabling interrupts, lock variables, strict alternation, Peterson’s Solution, TSL,* *semaphores, Mutexes, and spinlocks.*

**12. What are the advantages and disadvantages of busy-waiting and sleep-and-wake approaches for mutual exclusion? (10 point)**

*Busy-waiting*

*Advantages:*

*1. Simple to implement*

*2. Less overhead if the wait time is short, no context switch*

*Disadvantages:*

*1. Possible waste of CPU cycles which otherwise can be used by others*

*2. Possible priority inversion problems*

*Sleep-and-wake*

*Advantages:*

*1. No wasted CPU cycles*

*2. No priority inversion problems*

*Disadvantages:*

*1. More overhead due to context switches*

*2. Error-prone*

**13. Discuss the goals of CPU scheduling on different computer systems, e.g., batch systems, interactive systems and real-time systems. (5 point)**

*Batch systems:*

*1. Maximize throughput*

*2. Minimize turnaround time*

*3. Increase CPU utilization*

*Interactive systems:*

*1. Minimize response time*

*2. Meet users’ expectations in regard to proportionality*

*Real-time systems:*

*1. Meeting deadlines*

*2. Provide predictable performance*

**14. Assume that the following processes are to be executed on a uniprocessor system.**

**Based on their arrival time and CPU burst, calculate the average turnaround time and response time of these processes under the following scheduling policies: (15 point)**

**a. FCFS**

**b. Round Robin (quantum = 4 and 6)**

**c. Shortest Job First (preemptive and non-preemptive)**

**Process Arrival Time CPU burst**

**P1 0 12**

**P2 0 3**

**P3 2 7**

**P4 3 5**

**Compare the performance of above policies.**

*a. FCFS-1 (P1 first):*

*Average turnaround time: ((12-0)+(15-0)+(22-2)+(27-3))/4=17.75*

*Average response time: (0+(12-0)+(15-2)+(22-3))/4=11*

*FCFS-2 (P2 first):*

*Average turnaround time: ((3-0)+(15-0)+(22-2)+(27-3))/4=15.5*

*Average response time: (0+(3-0)+(15-2)+(22-3))/4=8.75*

*b. Round Robin (quantum=4)*

*Time: 0 4 7 11 15 19 22 23*

*Process: P1 P2 P3 P4 P1 P3 P4 P1*

*Average turnaround time: ((27-0)+(7-0)+(22-2)+(23-3))/4=18.5*

*Average response time: ((0+(4-0)+(7-2)+(11-3))/4=4.25*

*Round Robin (quantum=6)*

*Time: 0 6 9 15 20 26*

*Process: P1 P2 P3 P4 P1 P3*

*Average turnaround time: ((26-0)+(9-0)+(27-2)+(20-3))/4=19.25*

*Average response time: ((0+(6-0)+(9-2)+(15-3))/4=6.25*

*c. SJF (preemptive)*

*Average turnaround time: ((3-0)+(8-3)+(15-2)+(27-0))/4=12*

*Average response time: (0+(3-3)+(8-2)+(15-0))/4=5.25*

*SJF (non-preemptive)*

*Same as the preemptive version.*

*Comparison:*

*Best average turnaround time: SJF*

*Best average response time: RR with quantum=4*

*Worst average turnaround time: RR with quantum=6*

*Worst average response time: FCFS (P1 first)*

*Overall SJF performs best in this data set as it has the best avg. turnaround time and good avg. response time.*