OPERATING SYSTEM QUESTIONS

Student’s Name

Course

Professor’s Name

Institutional Affiliation

City

Date

Operating System Questions

**Question 1: Function and System Calls**

**Describe how function calls work**

Function calls are invocation that passes control to a subroutine, where upon execution the control returns to the next instruction in the main program. Function calls work by pushing all the parameters or arguments for the function, then the program issues a call instruction, indicating which function to be invoked. When a function is called by the main function, it redirects the control to another location, by making a space on the stack, where return address is stored.

**Describe how system calls work**

System calls is a mechanism where programs provide interaction with the operating system. Additionally, the interface between a process with an operating system, is facilitated by system calls (Edition, n.d, p.64-82). In terms of how system calls work, first the program arguments for the user application up, then a system call instruction is issued by the operating system, which causes the processor to jump to a specific address to start executing the code

**Difference between function calls and system calls**

 A function call involves an invocation to a subroutine in a program, whereas a system call involves an invocation of a subroutine that has been built in a system. Additionally, function calls are used when calling a specific function in a program, while systems call are primarily used when there is a need for a program to interact with a kernel.

**Question 2: Process and Threads**

**Describe what is a process**

A process is an instance of program in execution. Each process is assigned a unique ID when created, and referencing of such a process is done using the same ID. Another definition of a process is execution of a program that operates in that program.

**Describe what is a thread**

A thread is smallest unit of execution, which encompasses sequential flow within a program. When multiple threads are running simultaneously, the concept is referred to as multithreading.

**Which resources might be switched between processes that need not be switched for threads?**

Process and thread switching are handled in a through a computing known as context switching. Context switching is a concept that involves storing the state of processes or threads (Sawalha, Tull & Barnes, 2013, p.1-3). One of the main distinctions between thread and process switching is that, in thread switching, the virtual memory space resource, remains the same, but in process switching, the virtual memory address space is subjected to change.

**Does the difference between process and thread matter for scheduling on server with lots of users and lot of processes like login.ifo.uio.no? Please provide argument for your view.**

In thread scheduling on a server with a lot of user, the distinction comes in due to inter-process communication in process which is very slow, whereas inter-thread communication is faster because of sharing of memory. The rationale behind this is because each process running in the server with many users has its own memory space while when it comes threads, they use the memory of the process that they belong to

**Question 3: Scheduling**

**How do you design the scheduler for this operating system?**

In designing the scheduler for the operating system in the case under consideration, it is important to note that the scheduler can be implemented to keep the computing resources busy. There are six fundamental criteria to be followed, when designing the scheduler for the operating system, which include CPU utilization, throughput, response time, turnaround time, waiting time and fairness of the scheduler (Goel & Garg, 2013, p.246). The scheduler must be built on the above criteria, to support complex simulation processing by the operating system.

**Discuss whether preemptive and non-preemptive scheduling is appropriate for this system**

Preemptive scheduling is applied when a process switch from the running to the ready state or from waiting to ready state. On the other hand, non-preemptive scheduling is applied when a process terminates or switch from the running to waiting state. Both preemptive and non-preemptive are appropriate for the operating system dealing with complex simulations, as provided in the case study (Goel & Garg, 2013, p.247). Preemptive and non-preemptive scheduling can be developed within the system to employ the performance of the system even in complex programming, where priority and single monitoring applications is not an issue.

**Explain for 4 Scheduling Algorithms how they work and discuss how well each of them is suited for this task.**

Scheduling algorithms provides set of rules that are paramount what task is to be executed at a particular moment. The following four scheduling algorithms are suitable for the scenario provided.

*First Come First Serve (FCFS)-* This is the simplest scheduling algorithm which involves scheduling of resources to a process based on their arrival time. In this case, a process that request the CPU resource first, is the one allocated first. Based on the case study provided, since it is clearly indicated that priority and single monitoring of the applications is not an issue, then this algorithm can fit well.

***Multi-level Feedback Queue Scheduling-* This form of scheduling algorithm is meant to allow the process to move between the queues. In this case, the idea is to have separation of processes based on the characteristics of operating system, specifically CPU (Wang, Gong & Kastner, p.235-236). With complex simulation scenario, where no priority or monitoring of applications is not paramount, if a process utilizes too much CPU time, it is moved to lower priority queue.**

***Shortest Job Remaining Algorithm*- This is a preemptive scheduling algorithm, where the processor is assigned the are closest to completion. However, preemption can be done by the newer jobs with the completion time. In this case, this algorithm is ideal to prevent one process running indefinitely, while there are other processes with the shortest remaining time in the ready queue.**

***Round Robin Scheduling*- In this form of scheduling algorithm, each process is allocated a fixed amount of time known as quantum, then one a process is executed for the allocated time, it is preempted and other processes are executed. The whole process is repeated until all the processes are executed completely. This scheduling is suitable for the given task since context switching is involved, and it can save the states of all the process in complex simulations environment, where priority scheduling is a not an issue.**

**Propose one of the 4 scheduling algorithms and explain your choice**

**The proposed algorithm based on the scenario is *Multi-level Feedback Queue Scheduling.* The rationale behind this choice is because in complex simulations, multiple queues must be maintained for the intensive that have the common characteristics. The kind of operations in multi-level feedback algorithm is related to CPU bounds operations in one queue and all I/O bound jobs another job.**

**Question 4: Virtual Memory**

**Explain in general terms how virtual addresses are translated to physical addresses when multi-level paging is used……**

 **The concept of a virtual address to a physical address is known as address translation. Based on this question, it is important to note that virtual and physical addresses are 32 bits long, which limits the size of the two. Translating virtual addresses and physical addresses involves checking whether the virtual address is larger in size than the limit that has been setup, if the size is larger than memory management unit provides an exception. I the size of the virtual address is not larger than the set limit i.e. 32, then memory management unit adds the base address to the virtual address to produce the required physical address. The concept of memory looks up page in this case is provided by the memory management unit. Additionally, acceleration of the translation mechanism where fast look-up hardware cache is appended to close the central processing unit. The concept of paging is very paramount in the whole issue, because it requires copping up programs into pages, where each page is loaded to the memory on demand (Li et al., 2016, p.9). During the page replacement, the old pages that have been modified in the main memory need to be copied to the virtual memory before replacement is done, the CPU tracks the pages that have been updated by maintaining the dirty bit for every page. Upon updating in main memory, a dirty bit is set, then the dirty page created is copied to the virtual memory and then replaced.**

**With the given page size, how many levels must your virtual address translation at least have**

*Solution*

Q`1A 32-bit address can address 2^32 bytes in a byte in a byte addressable machine. Therefore, the size of the page is 1 Kilobyte (2^13,), the number of addressable page levels= (2^32)/ (2^13) =2^19> In this case there can be 2 levels of the address since the virtual address space is suing 32 bits. Additionally, having more levels can be cumbersome for the large address spaces, like 64 bits.

**Question 5: Splitting an Address Block into Networks**

**What is the netmask of the address block that has been assigned to your company?**

1024 IPV4 Address= Number of hosts will be given by 210 -2=1022

If 256 addresses =1 host, what about 1024? This will be given by 4 hosts.

The default netmask for 1 host with 256 addresses is given by 255.255.255.256. Therefore, in the given scenario, netmask will be given by (255.255.255.(256-4)) = **255.255.255.252**

**Netmask=255.255.255.252**

**How many addresses do you allocate to each of the networks, netA, netB and netC. What is the network address and the netmask for each of these networks? Explain how you get to this solution**

It is known that the first IP address is 9.239.16.0, each IP address can create 254 IP’s, which can allow 256 x 4=1024. Since each location requires

|  |  |
| --- | --- |
| Total Number of Hosts: | 1,024 |
| Number of Usable Hosts: | 1,022 |
| Binary Subnet Mask: (255.255.255.252) | 11111111.11111111.11111100.00000000 |
|

|  |  |
| --- | --- |
| Usable Host IP Range: | 9.239.16.1 – 9.239.19.254 |

Therefore, each of the three networks require 254 addresses |  |

The solution is obtained by identifying the number of Ips that can be created from the valid hosts. This is done upon identifying the subnet mask for the three networks

**Explain why you have chosen to divide your address range into three networks like this. Did you keep any addresses in reserve? And if yes, how do you plan to use them when necessary?**

The addresses are within a range of 254 in each network. The range is beyond the number of IP addresses required in each network. Therefore, those Ips that are remaining, can be kept in reserve to accommodate future expansion of each of the network.

**Question 6: Packets over TCP**

**Explain what the statement TCP provides a byte stream service means**

This concept means that bytes are written by the sender into TCP connection/ link and then the bytes are read by the receiver from the TCP connection. Additionally, TCP byte stream describes the concept of the services offered by the TCP.

**What does the byte stream service mean for applications that use socket function …….?**

In the applications that use socket function byte stream service is all about flow control for each byte transmitted via the TCP connection. Additionally, byte stream in this case is all about the service offered by TCP to application processes. Understandably, transmission control protocol from the source host buffers adequate bytes from the sending processes to fill the reasonable sized packets and then send the packets to their peer on the destination host (Mundra & El Taeib, 2015, p.416). In this case, the function send () and recv() are used while sending and receiving data from the source to the destination.

In socket programming, the programmers must use TCP protocol due to optimization and capacity for accurate delivery. Therefore, in real time applications that require socket programming, the programmers can take advantage of this. Moreover, TCP is meant for reliable stream delivery service, which provide a guarantee that the bytes received are identical and in the order in which they are sent.

**If an application uses TCP but needs packets, how must the developer design the application layer protocol**

Since TCP is known for providing reliable packets delivery service, with end to end error detection & correction, application programmers need to choose the best service, which is more appropriate for the specifications of the applications. In TCP data is viewed as a continuous stream of bytes but not as packets that are independent (Sultana & Wahid, 2019, p.5). Hence, application programmers can switch to UDP since its s able to cater for small chunks of data during transmission. Furthermore, the overhead of connections creation ensures reliable delivery, and retransmission of data sets when need arises. The developer must design the application layer protocol by mapping the application and network, because the socket provides the programming interface, where internet applications can be built in.

**Would it be easy to create a new transport control protocol Packet-TP with small changes to TCP that delivers a connection-oriented, reliable, end-to-end datagram service? List at least three required changes and explain why they are important. The changes can be small.**

Creating a transport control protocol is a tricky task, since getting it right is quite hard. Additionally, change the circumstances is what makes the whole process complicated. Most importantly, some specific changes are not hard to predict. In some cases, transport control protocol can be tweaked to deal with the upcoming circumstances as it is the case here (Dunkels & Vasseur, 2010, p.25-29). Ideally, making some changes to a protocol that is existing, to meet the requirements within its environment or the needs of the new applications might require tweaking the existing protocol rather than creating a new one, which might be costly.

In terms of the required changes they include transport-level services, changes in the applications requiring RPC semantics, and changes requiring the use of RTP. Such changes are important since this turns out to be simpler approach in meeting the efficiency of protocol and making the transport control protocol to adapt to the changes withing the environment.

**Question 7: Flow Control**

**Explain how the flow control mechanism selective repeat work**

The flow control mechanism selective repeat work especially when the network connection is unreliable. In this case, retransmission occurs frequently, where the selective frames retransmission becomes much efficient than retransmitting all the frames. For the selective repeat to occur, full duplex communication media is required for transmitting the data packets consistently, and then issue acknowledgment.

**Explain how flow control with a credit mechanism work.**

 Credit-based flow control is an efficient way for implementing flow-controlled networks via virtual circuit, which make use credit-based link by link. This mechanism work by counting buffer slots that are available at the downstream site, the count includes the arriving credits. Once new buffer slots are made available, the credits that corresponds are then sent upstream.

**Describe the differences between selective repeat and credit mechanism …….**

 Selective control mechanism involves sending specific number of frames at a time then wait for the acknowledgement to be received, for the packets transmitted a specific order. In a situation where packets are not received, the lost packets are retransmitted selectively, based on the acknowledgement of the receiver, the buffer of the lost packets is maintained by the receiver. On the other hand, credit flow control mechanism involves downstream to upstream notification, which is implemented using a credit record, where a virtual circuit connecting the remote nodes keeps on counting the buffer slots on the downstream, and then match them with the available slots on the upstream. The two mechanism handles the packet and ACK losses in different ways because selective controls only look at the packets not received and then they are retransmitted selectively, while in credit flow mechanism use the concept of virtual circuits to record the packets that have not been transmitted during the downstream and upstream. The size of the sequence number varies depending on the number of packets and the network traffic

**Can you find the problem with the loss of ACKs in the credit mechanism that does not exist in selective repeat, and do you have a proposal for solving it?**

In credit mechanism, the measurement of offline loss to control the transmission of packets can be implemented using cumulative coded acknowledgement, which is an accurate way of measuring the lost packets with accuracy (Qin, 2016, p.64. The proposed way for solving this is by using CCAK, which is a mechanism oblivious to the rate of loss, which is essential in allowing the nodes network-coded traffic to be acknowledged nodes to the upstream nodes in a simpler way.

**Question 8: Dijkstra’s Shortest Path First Algorithm.**

**Explain how Dijkstra’s Shortest Path Routing Algorithm Work. List of steps.**

The essence of Dijkstra’s algorithm is to compute the shortest between the one node and others in the graph.

*Steps in Dijkstra’s Shortest Path Routing Algorithm*

1. The first step in Dijkstra’s algorithm involves initializing the distances,
2. The second step involves picking the first node and calculating distances to the adjacent nodes
3. The third step is all about picking the node with minimum distance, the whole process of calculating the adjacent nodes is repeated.
4. The last step is to derive the final result of the shortest path tree.

The above steps are important as they help in finding the shortest path between two vertices in a graph that is weighted. Additionally, the algorithm is helpful in finding the shortest distances as well as the minimum costs thus becoming a valuable tool

**Use the algorithm to compute the distance from A**



If A is the source node and J is the destination node then

*Dijkstra’s Shortest Path Routing Algorithm* will be given by defining the possible paths, and then compute the one with the shortest distance as follows.

Possible paths include:

A-C-D-I-J = 9

A-B-E-I-J = 7

**A-B-E-G-H = 5**

A-B-F-H-G-J = 8

Based on the distances above, the *Dijkstra’s Shortest Path Routing Algorithm* path will be given by **A-B-E-G-H** with the smallest distance of 5

**Creating 2 Permanent labels right after F and right after E (see the table in the diagram below)**

**Finally, make a table that shows A’s routing table, consisting three columns i.e. Destination Node, Next node on the path to destination and Length of the shortest path**

Below is the routing table:

|  |  |  |
| --- | --- | --- |
| **Destination Node** | **Next node on the path to destination** | **Length of the shortest path** |
| A | A | 0 |
| B | B | 1 |
| E | E | 2 |
| G | G | 1 |
| J | J | 1 |

Reference List

Dunkels, A. and Vasseur, J.P., 2010. *Interconnecting Smart Objects with IP*. Morgan Kaufmann.

Edition, E., (n.d). The William Stallings Books on Computer.

Goel, N. and Garg, R.B., 2013. A comparative study of cpu scheduling algorithms. *arXiv preprint arXiv:1307.4165*.

Li, T., Zhang, J., Ma, K., Xiao, L. 2016. Virtual and physical address translation mechanism of interconnect network. *Journal of Beijing Institute of Technology*, (3), p.9.

Mundra, S. and El Taeib, T., 2015. TCP/IP protocol layering. *International Journal of Computer Science and Information Technology Research*, *3*(1), pp.415-417.

Sawalha, L., Tull, M.P. and Barnes, R.D., 2013. Hardware thread-context switching. *Electronics letters*, *49*(6), pp.389-391.

Sultana, T. and Wahid, K.A., 2019. Choice of application layer protocols for next generation video surveillance using Internet of video things. *IEEE Access*, *7*, pp.41607-41624.

Qin, Y. ed., 2016. *Network Coding at Different Layers in Wireless Networks*. Springer.

Wang, G., Gong, W. and Kastner, R., 2008. Operation scheduling: algorithms and applications. In *High-Level Synthesis* (pp. 231-255). Springer, Dordrecht.