|  |  |
| --- | --- |
| What is the benefits of Multithreaded Processes ? |  |
| * **Responsiveness** : If one thread stop the other threads will continue * **Resource** **Sharing** : Shearing the RAM, code, file and data * **Economy** : because of shearing the resources, the cost of creation process is about 30 times more than cost of create thread * **Scalability** : in case of multicores or multiprocessor , it means tow threads of the same process can execute in the same time (**parall**) |  |

|  |  |
| --- | --- |
| What is challenges of multicores system ? |  |
| * Dividing activities * Balance * Data splitting * Data dependency * Testing and debugging |  |

|  |  |
| --- | --- |
| Explain how the multithreaded is suitable to client server architecture ? |  |
| With multithreaded if the client send request to the server to display web page content (that may be pictures, form, text, other), each item will be assigned to a thread, so if a thread of picture is stop the other parts of the web page will be displayed , but without multithreaded all web page content will not displayed. |  |

|  |  |
| --- | --- |
| What is the types of threads ? |  |
| * **User threads** : threads management dine by user level threads library with API **Like** (POSIX threads, Win32 threads, Java threads) * **Kernel threads** : supported by the kernel **Like** (WindowsXP/2000, Solaris, Linux, Mac OS X) * **The threads manage by thread libraries** |  |

|  |  |
| --- | --- |
| What is the model of multithreading ? |  |
| * **Many-to-one** : Many user-level threads mapped to single kernel thread   **Adv**: no overhead in kernel  **Dis**: if kernel thread stop the other thread will stop   * **One-to-One** : Each user-level thread maps to kernel thread   **Adv**: if kernel thread stop the other thread will continue  **Dis**: no overhead in kernel     * **Many-to-Many** : Allows many user level threads to be mapped to many kernel threads, and Allows the operating system to create a sufficient number of kernel threads   **Adv**: no overhead in kernel , and if kernel thread stop the other thread will continue   * **Tow Level** : Similar to many-to-many, except that it allows a user thread to be **bound** to kernel thread   **Adv**: allows a user thread to be **bound** to kernel thread |  |

|  |  |
| --- | --- |
| What is the types of threads library implementation ? |  |
| * **Exists in user space** : code and data structure for managing the thread in user space **example:** Java library threads * **Exists in kernel space** : code and data structure for managing the thread in kernel space **example:** Win32 thread library   **P thread library is example for both types (MAC-Linux)** |  |

|  |  |
| --- | --- |
| What of Java thread created ? |  |
| * Extending Thread class * Implementing the Runnable interface |  |

|  |  |
| --- | --- |
| Does **fork()** duplicate only the calling thread or all threads? |  |
| 1. If **exe()** come after **fork()** immediately duplicate just the caller thread 2. If not will duplicate all threads |  |

|  |  |
| --- | --- |
| What is thread cancelation? |  |
| Terminating a thread before it has finished |  |

|  |  |
| --- | --- |
| What is general approaches to cancel thread ? |  |
| * **Asynchronous cancellation** terminates the target thread immediately * **Deferred cancellation** allows the target thread to periodically check if it should be cancelled |  |

|  |  |
| --- | --- |
| What is cancelation point ? |  |
| The cancelation occur only after the target thread has checked a flag to determine either or not it should be canceled .  The thread can perform this check at a point at which it can be cancel (cancelation point in P thread). |  |