Notifier Chains

1. The notifier chain facility is a general mechanism provided by the Linux kernel.
2. Notifier chains are used to inform asynchronous events or status, through the function calls registered.
3. The notifier data structure is a simple linked list of function pointers. The function pointers are registered with 'functions' that are to be called when an event occurs. Each module needs to maintain a notifier list. The functions are registered to this notification list.
4. Notifier chains are broadly classified based on the context in which they are executed and the lock/protect mechanism of the calling chain. Based on the need of the module, the notifiers can be executed.
5. Notifier chains are classified into four types: Atomic, Blocking, Raw, SRCU.
6. Atomic notifier chains: This notifier chain is executed in interrupt or atomic context. Normally, events that are time critical, use this notifier. This also means it is a non-blockable call. Linux modules use atomic notifier chains to inform watchdog timers or message handlers.
7. Blocking notifier chains: This notifier chain runs in the process context. The calls in the notification list could be blocked as it runs in the process context. Notifications that are not highly time critical could use blocking notifier chains. Linux modules use blocking notifier chains to inform the modules on a change in QOS value or the addition of a new device.
8. Raw notifier chains: This chain does not manage the locking and protection of the callers. Also, there are no restrictions on callbacks, registration, or de-registration. It provides flexibility to the user to have individual lock and protection mechanisms. Linux uses the raw notifier chain in low-level events.
9. SRCU notifier chains: Sleepable Read Copy Update (SRCU) notifier chains are similar to the blocking notifier chain and run in the process context. It differs in the way it handles locking and protection. The SRCU methodology brings in less overhead when we notify the registered callers. On the flip side, it consumes more resource while unregistering. It is advisable to choose this methodology where we use the notifier call often and where there's very little requirement for removing from the chain.
10. The kernel's implementation of notifier chains is unsafe. There is no protection against entries being added to or removed from a chain while the chain is in use.