Signal flags are used in systems programming to represent events or conditions. When one part of a program needs to notify another that something has happened (like an interrupt or a state change), it can set a signal flag. The receiving part checks this flag to determine if it should act.

Think of a signal flag like raising your hand in class, it's a simple way to get attention without needing a full conversation.

Semaphores are synchronization tools used to control access to shared resources in concurrent programming (like in operating systems or multithreaded applications). A semaphore flag acts as a counter or gatekeeper:

* Binary semaphore (flag): Has two states (0 or 1) — used like a lock to allow or block access.
* Counting semaphore: Can count multiple resources — it decreases when a resource is taken and increases when released.

Semaphore flags help avoid issues like race conditions and ensure that only a certain number of threads access a resource at the same time.