Standard way: *monolithic kernel*:

Only two levels: user mode and kernel mode
All kernel code executed in kernel mode with full privileges

Example: Linux

Idea: Restrict amount of code running in kernel mode to minimum
⇒ Implement remainder of OS as services

At bottom: have microkernel with functions like

- Memory Management
- Scheduling
- Low-level device drivers

Higher-level parts like filesystems implemented in user space

Message passing used
Often combined with capabilities for good permission handling
⇒ Efficient message passing vital for performance

Message passing lends itself to asynchronous communication
⇒ bad for implementing Unix system calls
Suitable for embedded systems, in particular special real-time OS