Lecture 16: Kernels

kernel – smallest portion of operating system that provides for task scheduling, dispatching, intertask communication, synchronization, and concurrency

```
     user applications        => software programs and GUIs running on top of operating system (O/S)
\                          /
 \   operating system  /      => executive + user interface shell                          (interface mgt)
  \      executive      /     => kernel + memory storage and I/O                          (memory mgt)
   \       kernel       /     => micro-kernel + task synchronization/communication         (service mgt)
    \   micro-kernel  /       => nano-kernel + task scheduler                              (CPU mgt)
     \  nano-kernel  /        => task controller and dispatcher                            (process mgt)
      \___________/
    hardware layer            => clock, CPU, ROM, RAM, gates, interrupt registers, context switch logic
```

nano-kernel: does task bookkeeping, execution, storage, division of activities, and dispatching

micro-kernel: schedules nano-kernel tasks involving real-time issues (hard or soft), priority management, interrupt handling, and time management

kernel: provides message queues, mailboxes, semaphores, pipes, and sockets to micro-kernel

executive: provides memory and I/O services to the kernel
cyclic executive: executive run as a periodic process by the kernel
Note: Most commercial RTOSes these days are run as cyclic executives. They are efficient, simple, control other periodic processes, predictable, have fast context switching; but they create design contraints, subject to breaking if rescheduling (overhead) is needed. Alternatively can use foreground-background executive.

operating system (O/S): provides file security, file management, and user tool command interface to the executive; acts as an interface to the computer hardware below and the software/user applications above

real-time operation system (RTOS): O/S that allows user access to the low level system services and resources to ensure tasks are predictable, visible, schedulable, concurrent, and dependable:

predictable -tasks will execute within bounded time constraints and meet their assigned deadlines
visible -tasks can be managed by system services, memory management is available, interrupt and fault/exception handling are available, and I/O services are utilized
schedulable -tasks have time-sharing and priority allocation and access to CPU using static or dynamic priority handling management
concurrent -tools are provided for task communication and synchronization and resource sharing such as mailboxes, semaphores, message queues
dependable -tasks will meet their deadlines even if system faults occur

Other features:
-keeps list of task pointers for scheduling on the stack
-manages stack memory, cache, on-board memory
-compiler library support and optional debugging services
-provides floating point support
-is as small as possible (5 – 20 KB of ROM)
-has network management (e.g. FTP, SNMP)
-provides device driver support including I/O and other embedded hardware initialization
-does task management and scheduling
-provides task communication tools like mailboxes, message queues, global memory, semaphores
-provides clock, timing services, counter functions, internal interrupt handling
-allows external interrupt register access and polling
-enables interrupt priority levels
-allows creation and initialization of software system devices