SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)
COIMBATORE-35
Accredited by NBA-AICTE and Accredited by NAAC – UGC with A++ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

19EEE305 / EMBEDDED SYSTEMS
III YEAR / VI SEMESTER

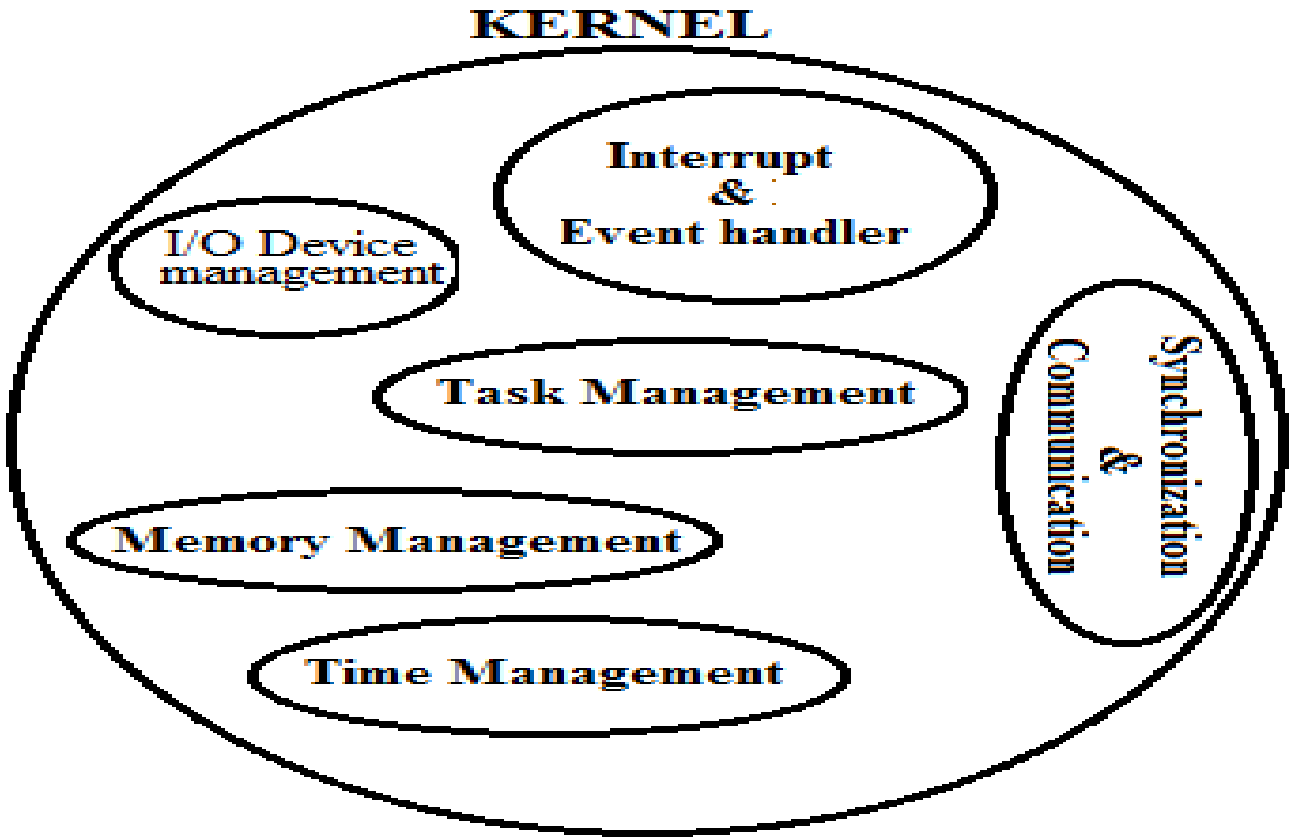
UNIT-IV: RTOS BASED EMBEDDED SYSTEM DESIGN

KERNAL

KERNAL ARCHITECTURE

Kernel is the core of an operating system

It is a piece of software responsible for providing secure access to the system's hardware and to running the programs



KERNAL

- It is responsible for scheduling running of user and other processes.
- It is responsible for allocating memory.
- It is responsible for managing the swapping between memory and disk.
- It is responsible for moving data to and from the peripherals.
- It receives service requests from the processes and honours them.

KERNAL

Let, At any one time we have *one process engaging the CPU. This* may be a *user process or a system routine (like ls, chmod) that is, providing a service.*

The following three situations result in switching to kernel mode from user mode of operation:

Situation1:

- The scheduler allocates a user process a slice of time (about 0.1 second) and then system clock interrupts.
- This entails storage of the currently running process status and selecting another runnable process to execute.
- This switching is done in kernel mode.

KERNAL

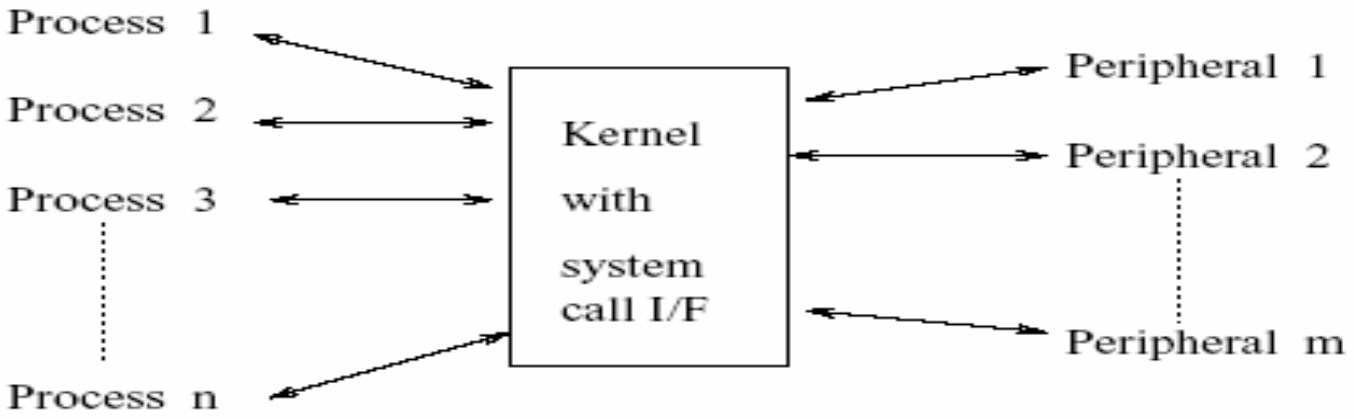
Situation2:

- Services are provided by kernel by switching to the kernel mode.
- So if a user program needs a service (such as print service, or access to another file for some data) the operation switches to the kernel mode.
- If the user is seeking a peripheral transfer like reading a data from keyboard, the scheduler puts the currently running process to “sleep” mode.

KERNAL

Situation 3:

- Suppose a user process had sought a data and the peripheral is now ready to provide the data, then the process interrupts.
- The hardware interrupts are handled by switching to the kernel mode.
- In other words, the kernel acts as the via-media between all the processes and the hardware as depicted in the below

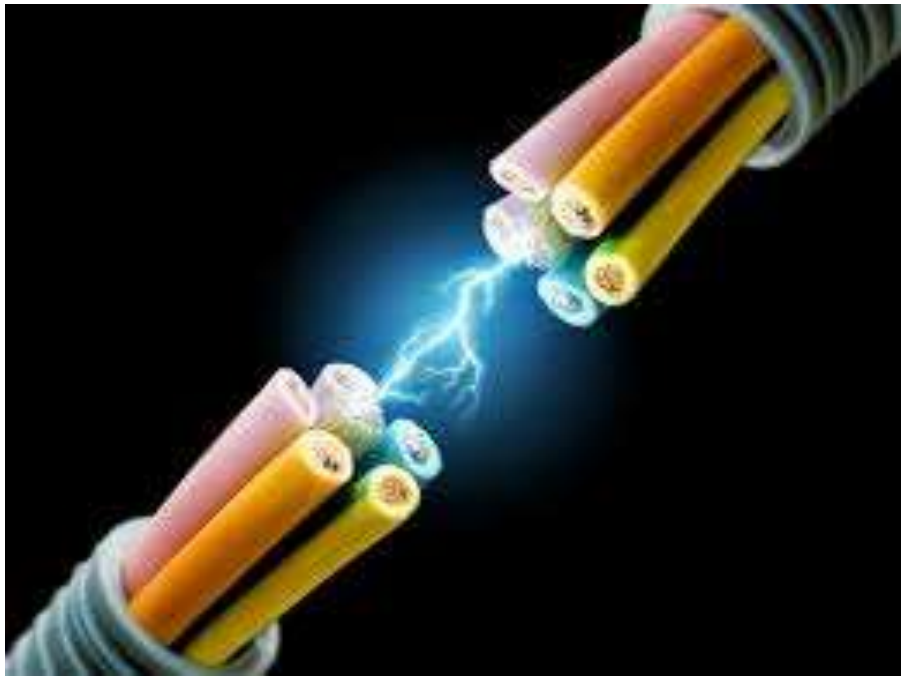


KERNAL PROCESS

Typically, Unix kernels execute the following secure seven steps on a system call:

- 1. Arguments (if present) for the system call are determined.
- 2. Arguments (if present) for the system call are pushed in astack.
- 3. The state of calling process is saved in a user structure.
- 4. The process switches to kernel mode.
- 5. The syscall vector is used as an interface to the kernel routine.
- 6. The kernel initiates the services routine and a return value is obtained from the kernel service routine.
- 7. The return value is converted to a *c version (usually an integer* or a long integer).
- The value is returned to process which initiated the call. The system also logs the user id of the process that initiated that call.

RECAP….



…THANK YOU