Explain the key concepts of Real Time Operating Systems.

M4.01 Explain the functions of an Operating System,
M4.02 Summarize the features of different types of Operating Systems.
M4.03 Outline key concepts of Task, Process and Threads.
M4.04 Explain multiprocessing and multitasking.
M4.05 Outline the key features of Task Scheduling algorithms.
M4.06 Summaries the key concepts of Task Communication and Synchronization
M4.07 Explain Device Drivers
M4.08 List the functional and nonfunctional requirements in selecting a RTOS

Contents:
Real Time Operating Systems (RTOS) – OS Basics, Types of OS, Process, Task and Threads, Multiprocessing and Multitasking, Task Scheduling, Task Communication, Task Synchronization, Device Drivers, How to choose RTOS.

Real Time OS

- OS act as a bridge between the user application and system resources.
- Primary functions are:-
  - Make the system convenient to use.
  - Organise and manage the system resources efficiently and correctly.

Functions of OS

- **Process management**-includes setting up memory space for process, loading code to memory,allocating system resources,scheduling and managing the execution of process, scheduling and managing PCB, inter process communication, process termination etc.
- **Memory management-**includes keeping track of memory area is currently used by the process,allocating and de-allocating memory space.
- **File system management**-includes create/delete/alteration of files and directories,naming convention, automatic allocation of file space,save a file etc.
- **Device management**-loading and unloading of device drivers, exchanging information to and from the device.

kernel

- Core of OS
- Responsible for system resources and communication among the hardware and other system services.
- Functions of kernel is same as functions of OS.
- Also provides protection of system, and handler mechanism for all external/ internal interrupts generated by the system.
- Kernel space-memory space at which kernel code located.it is protected area.
- User space- memory space at which user programs located.
- Types-monolithic and micro

monolithic

- all kernel modules run within the same memoryspace under a single kernel thread.
- Adv-effective utilisation of the low level features of underlying system
- Disadv-any failure in any one of the kernel module leads the crashing of entire kernel application.
- Example- linux,solaris,ms-dos

Micro kernel

- Incoporates only essential set of OS services to kernel. Other services runs in user space.
- Microkernel handles memory management, process management, timer sytem and interrupt handlers.
- e.g.Math,QNX,Minix 3 kernel
- Adv-robustness, configurability

OS types

- GPOS(General Purpose Operating System)
  - OS used in general computing system
  - Non-deterministic
  - e.g.windows/MS-DOS etc.
- Real Time OS(RTOS)
  - OS designed for devices that are timing specific.
  - Deterministic
  - e.g. windows CE,QNX etc.

Task, Process and Threads

Task

- Is the program in execution
- Also known as job

Process

- Also known as instance
- a program in execution
- contains one or more threads
- it has its own data, code and stack memory
- it contains atleast one thread
- expensive to create

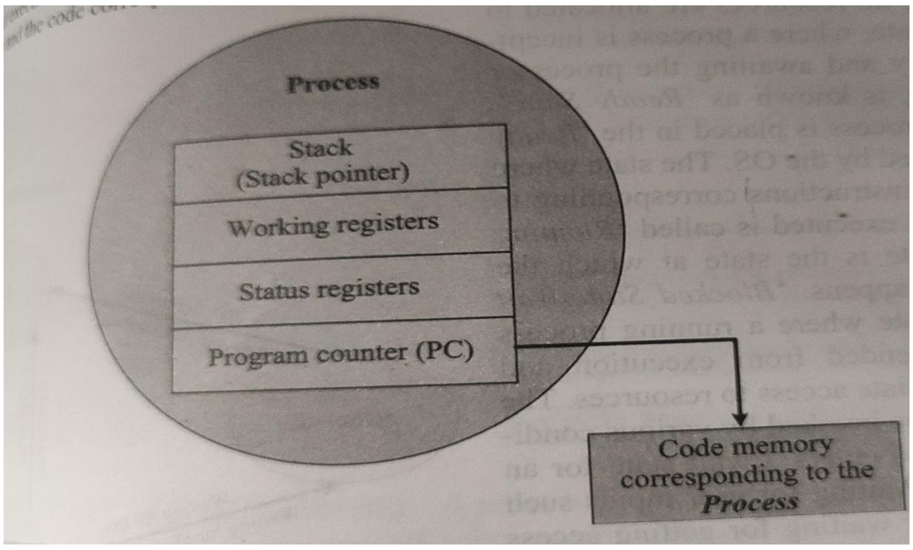
Threads

- Also known as light weight process
- single unit of execution
- is a part of process
- it shares the memory with other threads of the same process
- can not live independently
- there can be muliple threads in a process
- inexpensive to create

Process

structure of process

- the concept of process is concurrent execution.
- It is achieved through the sharing of CPU among the process.
- A process mimics a processor in properties.it holds-registers,status, program counter and stack.
-



Memory organisation of process

- memory occupied by the process is divided into 3-stack memory, data memory, code memory.
- Stack memory holds temporary data, data memory holds the global data for the process,code memory holds the program code for the process.
- On loading a process into a memory, a specific area of memory is allocated for the process.

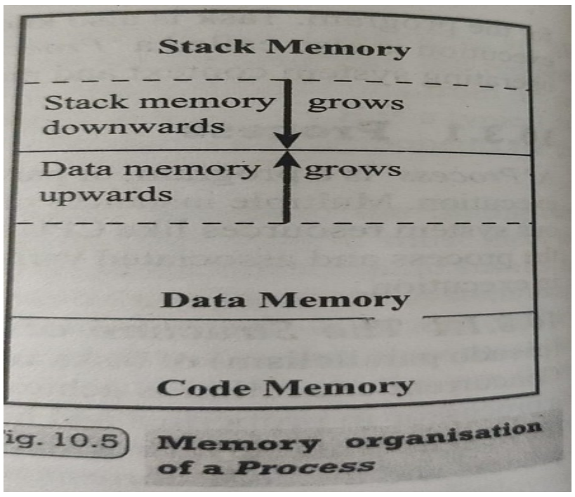
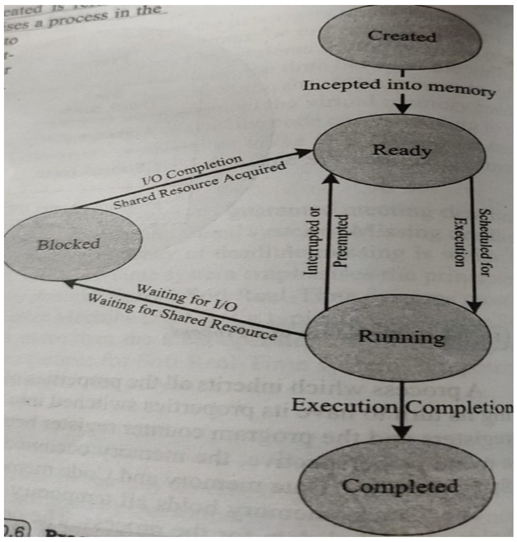


Fig. 10.5 Memory organisation of a *Process*

Process states

- the cycle throug which a process changes its state from 'newly created' to 'execution completed' is known as "process life cycle".
- Process states are:-
  1. created -the state at which a process is being created. This state no resources are allocated to the process.
  2. Ready state-the state at which the process is placed in the ready list
  3. Running state-the state at which the process execution happens.
  4. Blocked /Wait state- a state where a running process is temporarily suspended from execution and does not have immediate access to resources.
  5. Completed state-a state where the process completes its execution.



Process management- it deals with the creation of a process, setting up the memory space for the process,loading the code, allocating resources and setting up a process control block(PCB)

Threads

- it is a single sequential flow of control within a process.
- It is also known as light weight process.
- It can have many threads.
- Different threads, which are part of a process, share the same address space.
- Threads maintain their own thread status, program counter and stack.

Multithreading-

- if a process is split into multiple threads, there will be a main thread and rest of the threads will be created within the main thread.
- The thread execute a portion of a process.
- Adv :
  - better memory utilisation-multiple threads of the same process share the address space for data

    memory.
    - Speed up the execution of the process.
    - Efficient CPU utilisation.

Thread standards

- different standards are available for thread creation and management.
- It is a set of thread class libraries.
- Commonly available thread class libraries are:
    - POSIX threads- Portable Operating System Interface. Standard library is Pthreads
    - win32 threads-used in windows os. These threads are created with the API.
    - Java threads- threads supported by the java programming language. Thread class is Thread and in library is java.lang.

Explain Multiprocessing and Multitasking

Multiprocessing

- it is the ability to execute multiple processes simultaneously.
- Systems which performs multiprocessing is known as multiprocessor system.
- system have multiple CPU.

Multiprogramming

- the ability of the OS to have multiple programs in memory and ready for execution.

Multitasking

- it is the ability to hold multiple processes in memory and switch the processor from one process to another.
- Tha act of switching CPU among the processes is known as context switching.
- Types-depending on switching
    - **co-operative**-any process can hold the CPU as much time as it wants.process gets a chance to execute only when currently executing process relinquishes the CPU.
    - **preemptive**-currently running process is preempted to give a chance to other process to execute. Every process gets a chance to execute.
    - **Non-preemptive-**the process is allowed to execute unti it terminates or enters the blocked/waited state.

Outline the key features of Task Scheduling algorithms.

Task scheduling algorithms

- to determine which process is to be executed at a given point of time is known as process/ task scheduling.
- The scheduling policies forms the guidelines for determining task to be executed when.
- Scheduling policies are implemented in an algorithm.
- A good scheduling algorithm has high CPU utilization, minimum Turn Around Time, maximum throughput and least response time.
- CPU utilisation-how much percentage of the CPU is being utilised. It must be high.
- Throughput -the number of process executed per unit of time. It must be high.
- Turnaround time-time taken by a process for completing its execution.it must be minimum.
- Waiting time-the amount of time spent in the ready queue to get the CPU for execution. It must be minimum.
- Response time-time elapsed between the submission of a process and the first response. It must be less.

Scheduling algorithms

Non- preemptive scheduling algorithms	**Preemptive scheduling algorithms**
Currently executing process is allowed to run until it terminates or enters the wait state.algorithms are:- 1. **First Come First Served/FIFO scheduling** allocates CPU time to the processes based on the order in which they enter the ready queue. First entered process is serviced first.draw backs-it favours monopoly of process. The average waiting time is not	Can stop temporarily the currently executing process and select another process for execution. Algorithms are:- 1.**preemptive SJF scheduling/shortest Remaining Time(SRT)-** if the execution time of new process is less, the currently executing process is preempted and new

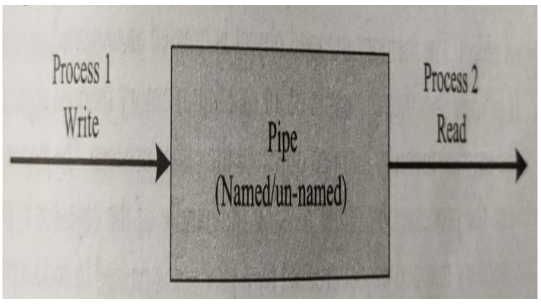
minimal. 2. **Last Come-First Served(LCFS)/LIFO scheduling:-** allocates CPU time to the processes based on the order in which they enter the ready queue.last entered process is serviced first. Drawback is same as FCFS. 3. **shortest Job First Scheduling(SJF):-** the process with the shortest estimated run time is scheduled first, followed by the next shortest path.the average waiting time for a given set of process is minimal.drawback-starvation,it is difficult to know in advance the next shortest process in ready queue. 4. **priority Based Scheduling:-** a process with high priority is serviced first. While creating a process or task , the priority can be assigned to it. Drawback- starvation-a process whose priority is low may not get a chance to execute. The technique to prevent starvation is known as Aging.	process is scheduled for execution. This scheduling algorithm always compares the execution completion time of a new process with the currently executing process and schedule with shortest remaining time for execution. 2. **Round Robin Scheduling:-** each process in the ready queue is executed for a predefined time slot. The excution start with the the first process, execute a predefined time and when the time/ process completes, the next process is selected for execution. Every process gets a fixed amount of CPU time for execution. If the process completes before the predefined time, process release the CPU and next process scheduled for execution. The implementation is kernel dependent. It is very difficult to maintain time slice for every process. **3.Priority Based Scheduling** RTOS uses preemptive priority based scheduling algorithm. In this, any high priority process entering the ready queue is immediately scheduled for execution. In non preemptive priority based scheduling,any high priority process entering the ready queue is scheduled only after the currently executing process completes its execution.

Summaries the key concepts of Task Communication and Synchronization(4)

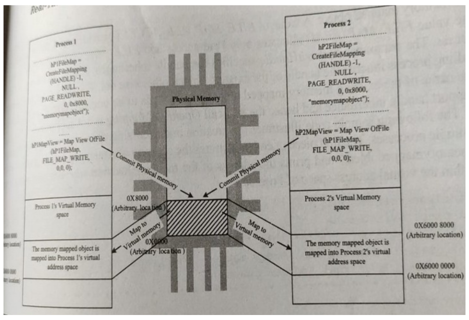
Task communication

- the mechanism through which processes/tasks communicate with eachother is known as inter process/task communication(IPC).
- It is essential for process co-ordination.
- It is OS dependent.
- Some mechanisms are:-

  1. shared memory
  - processes share some area of memory to communicate with eachother.
  - Information to be communicate by the process is written in shared memory.
  - A lots of data can be shared.
  - It is slow method.
  - Implementing mechanisms are:-
  - **1.pipes**
    - it is a section of shared memory. It follow client-server architecture.

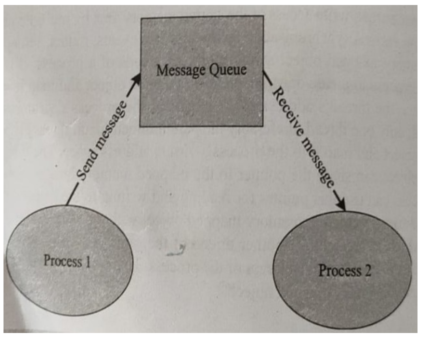


- 2. **memory mapped objects**
  - it is a shared block of memory.

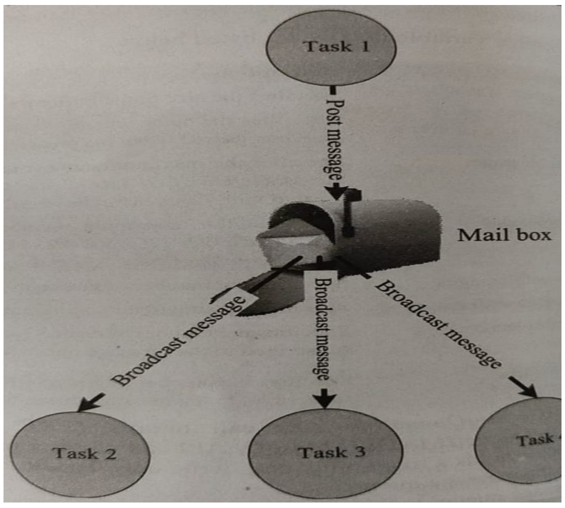

Process 1
hP1FileMap = CreateFileMapping (HANDLE) -1, NULL, PAGE_READWRITE, 0, 0x8000, "memorymapobject");
hP1MapView = Map View OfFile (hP1FileMap, FILE_MAP_WRITE, 0,0, 0);
Process 1's Virtual Memory space
The memory mapped object is mapped into Process 1's virtual address space
Physical Memory
Commit Physical memory
Commit Physical memory
0X8000 (Arbitrary location)
0X0000 (Arbitrary location)
Map to Virtual memory
Map to Virtual memory
Process 2
hP2FileMap = CreateFileMapping (HANDLE) -1, NULL, PAGE_READWRITE, 0, 0x8000, "memorymapobject");
hP2MapView = Map View OfFile (hP1FileMap, FILE_MAP_WRITE, 0,0, 0);
Process 2's Virtual Memory space
The memory mapped object is mapped into Process 2's virtual address space
0X6000 8000 (Arbitrary location)
0X6000 0000 (Arbitrary location)

2. message passing

- it is a synchronous information exchange mechanism.
- Limited amount of data can be shared.
- It is fast .
- Implementation:-
- **1. message queue**
  - it stores messages temporarily in a system. It is implemented through send and receive methods.



- **2.mailbox**
  - used for one way messaging.the process which wants to send a message, creates a mailbox for posting the message.the process which are interested in receiving the message can subscribe the mailbox.

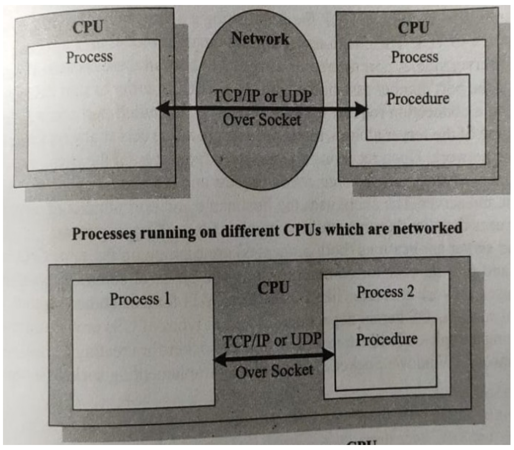


- **3. signalling**
  - it is used for asynchronous notification. One process wants to communicate with other, it send a signal to other.

3. remote procedure call(RPC) and sockets

- it is used for client- server applications.
- Also known as Remote Method Invocation.

- Used by a process to call a procedure of another process running on the same CPU or different CPU in a network.
- This communication can be synchronous or asynchronous.
- Sockets are used for RPC communication.
- Socket is a logical endpoint in a two way communication link between two application.



Task Synchronisation

- the act of making processes aware of the access of the shared resources by each process to avoid conflict is known as task synchronisation.
- Issues:-
  - 1. **racing**- it is the situation in which multiple processes compete each other toe access and manipulate shared data concurrently.
  - 2. **dead lock**-it creates a situation where none of the processes are able to make any progress in their execution.the different condition that deadlock occure-(also known as coffman conditions)
    - mutual exclusion-the criteria that only one process can hold a resource at a time.e.g. accessing of display hardware in an ES.
    - Hold and wait-a process holds a shared resource and waiting for additional resources held by other processes.
    - No resource preemption-OS can not take back a resource from a process which is currently holding it.
    - Circular wait-a process is waiting for a resource which is currently held by another process which in turn is waiting for a resource held by the first process.
    - Deadlock avoidance methods-
      - 1. ignore deadlocks. e.g.unix
      - 2.detect and recover-
      - 3.avoid deadlocks-careful resource allocation
      - 4.prevent deadlocks-

      livelock- a process in livelock changes its state with time.a process always does something but is unable to make any progress in execution completion.

      starvation-a process does not get the resource required to continue its execution for a long time.

3. **dining philosopher's problem**-five philosophers are sitting around a round table, involved in eating and brainstorming. A philosopher will be any one of three states- eating,hungry and brainstorming.for eating a philosopher requires 2 forks. There are only 5 forks.

4.**producer-consumer/bounded buffer problem**-this is a common data sharing problem where two processes concurrently access a shared buffer with fixed size.

5. **priority inversion-** a high priority task needs to wait for a low priority task to release a resource which is shared between the different priority processes.

Task Synchronisation Techniques

- the code memory area which holds the program instruction for accessing a shared resource is known as critical section.
- Inorder to synchronise the access to shared resources, the access to the critical section should be

exclusive.

- Task synchronisation techniques are:-
  1. **mutual exclusion through busy waiting/spin lock**-it uses a lock variable. Each process checks this lock variable before entering the critical section.
     If lock=1, the process is in its critical section.i.e no other process can use the shared memory. Otherwise lock=0, i.e the process can use the shared memory.
     Drawback-wastage of CPU time, high power consumption.
  2. **mutual exclusion through sleep/wakeup-**when a process is not allowed to access the critical section, then the process undergoes sleep state. When the process leaves the critical section,it send wakeup message to another processes which is waiting for the access to the critical section.**semaphore:-** it is a sleep/wakeup based mutual exclusion implementation method.

Explain Device Drivers

- act as a bridge between the OS and hardware.it abstracts the hardware from user applications.
- Role of device driver in embedded OS
- fig:

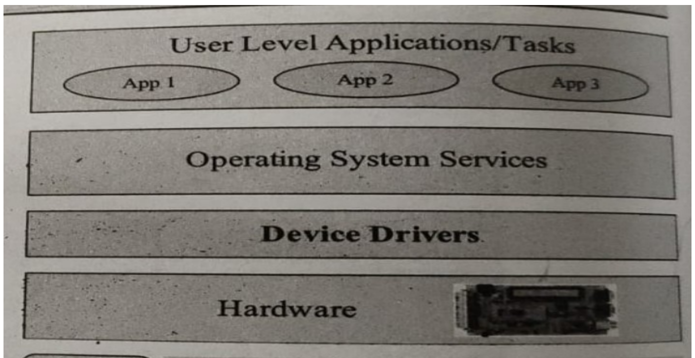


Fig. 10.39 Role of Device driver in Embedded OS based products

- device driver responsible for :
  - initiating and managing the communication with the hardware peripherals
  - establish the connection to the device
  - initialising the hardware and transfering data.
- Certain drivers are a part of the OS kernel(known as built-in driver/onboard driver) and certain drivers need to be installed on the fly(installable drivers).
- onboard drivers are loaded by the OS at the time of booting device and are always kept inthe RAM. Installable drivers are loaded by the OS on needed basis.
- A device driver implements the following:
  - device initialisation and interrupt configuration
  - interrupt handling and processing- i.e interrupt type,bind the interrupt with interrupt request,register an interrupt with interrupt request.
  - interfacing with user applications.

List the functional and nonfunctional requirements in selecting an RTOS

functional requirements

1. **processor support**
2. **memory requirements**- evaluate the minimal ROM and Ram requirements.
3. **Real-time capabiities-**
4. **kernel and interrupt latency**-kernel may disable interrupts while executing services and it may lead to latency.for an ES, latency should be minimum.
5. I**nter process communication and task synchronisation-** it is kernel dependent
6. **modularisation support**- developer can choose the essential modules and re-compile the OS image for functioning.
7. **Support for networking and communication**-os provides support for all the interfaces required by the embedded product.
8. **Development language support-**OS may include run time libraries as built in components

Non-functional requirements

1. **custom developed or off the shelf**-readily available OS which is in close match with the system requirements.
2. **Cost**-total cost for developing or buying the OS
3. **development and debugging tools avilability**-explore the different tools available for the OS.
4. **Ease of use**-how easy it is to use.
5. **After sales**-in the form of email/on call services,critical path updates and support for production issues should be analysed thoroughly.

Important Questions
1. what is process life cycle.
2. explain how threads and processes are related?what are common to task and processes?
3. explain multiprocessing, multitasking, and multiprogramming.
4. what is device driver?
5.explain the different functional requirements and non-functional requirements that need to be evaluated in the selection of an RTOS.
6.what is critical section?what are the different techniques for controlling access to critical section.
7. what are the key features of task scheduling algorithm.
8. explain the memory architecture of process.
9. differentiate process and threads.
10.what is an OS.where is it used and what are its primary functions?