REAL TIME OPERATING SYSTEM

DESKTOP VS. RTOS

Desktop OS	RTOS
During boot up, OS gets the control first then the application software.	During boot up, Application software gets the control first then the OS.
Application and OS share the different address space.	Application and OS share the same address space.
OS protects itself carefully from the application.	OS does not protect itself carefully from the application.
General Purpose.	Dedicated to a single embedded application.
Windows, Mac OS etc.	Vxworks, Vtrx, Nucleus, LynxOS etc…

INTRODUCTION TO RTOS

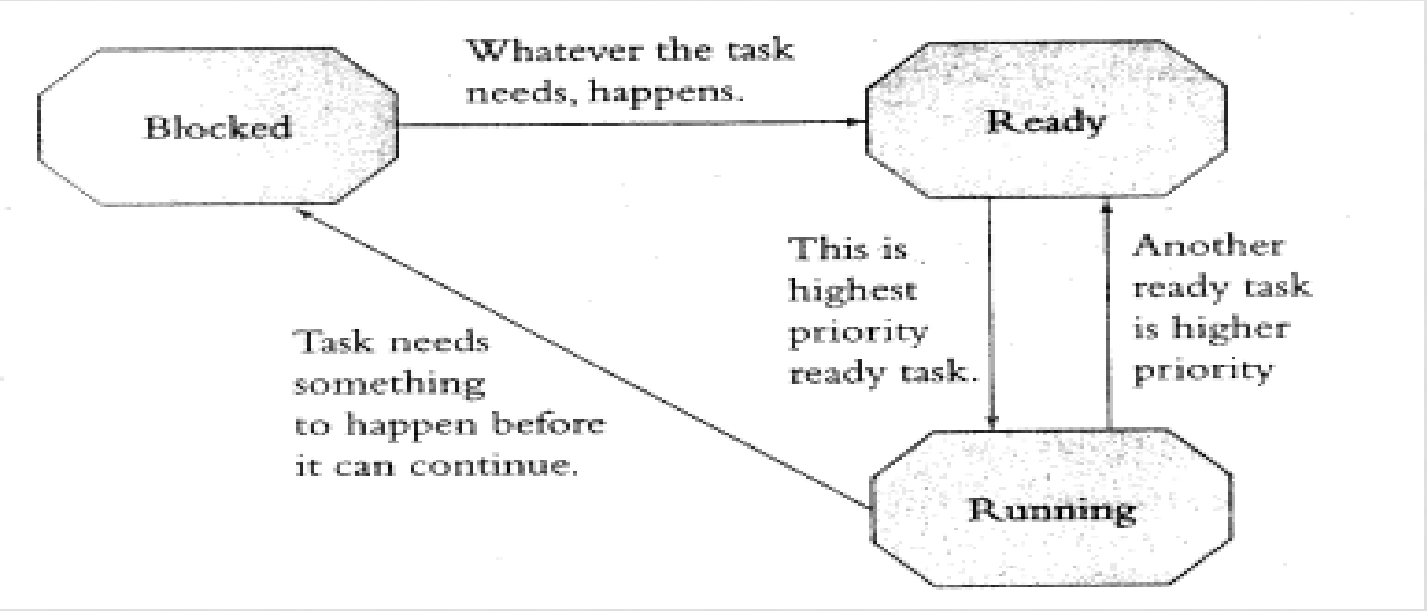
- **Reliability**
  - Systems must be reliable.
  - Needs to operate without human intervention.
- **Predictability**
  - Should be deterministic(meeting time requirements should be predictable).
- **Performance**
  - Fast enough to fulfill the timing requirement.
- **Compactness**
  - Should be small and efficient to face memory constraints.
- **Scalability**
  - Should be capable of adding/deleting modular components.

TASK AND ITS STATES

Task – a simple subroutine.

set of program instructions that are loaded in memory.

Task states: Running or Executing, Blocked(waited, dormant, delayed), Ready(pending, suspended).



TASK AND ITS STATES

Task = Code + Data + State.

Task State is stored in a TCB(Task Control Block).

TCB:

ID
Priority
Status
Registers
Saved PC
Saved SP

TASK AND ITS STATES

- **What happens if two tasks with the same priority are ready?**
- **If one task is running and another high priority task unblocks, does the task that is running get stopped and moved to the ready right away?**
- **What happens if all the tasks are blocked?**

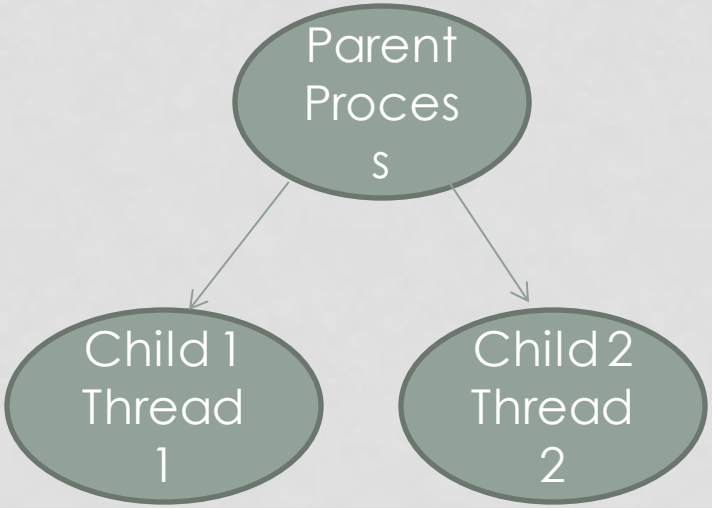
PROCESS AND THREAD

Each application has multiple process.

Each process has its own address space, cpu quota, access to hardware resources and kernel services.

Process is allocated memory for instruction and data.(process needs memory for code and data.)

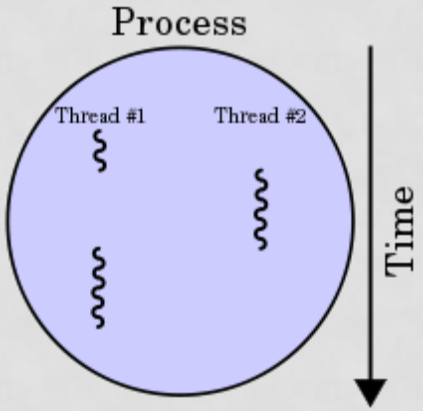
Example: Parent process is auto created during system start up. Thread is created dynamically.



PROCESS AND THREADS

Process

Process	Thread
Program in execution.	Function within executable.
Requires separate address space.	Shares same address space.
Inter Process communication is expensive	Inter thread communication is less expensive when compared to process.
Context switching from one process to other is costly.	Context switching between thread is less costly.
It has its own memory, program counter and stack.	It has shared memory.



REAL TIME KERNEL

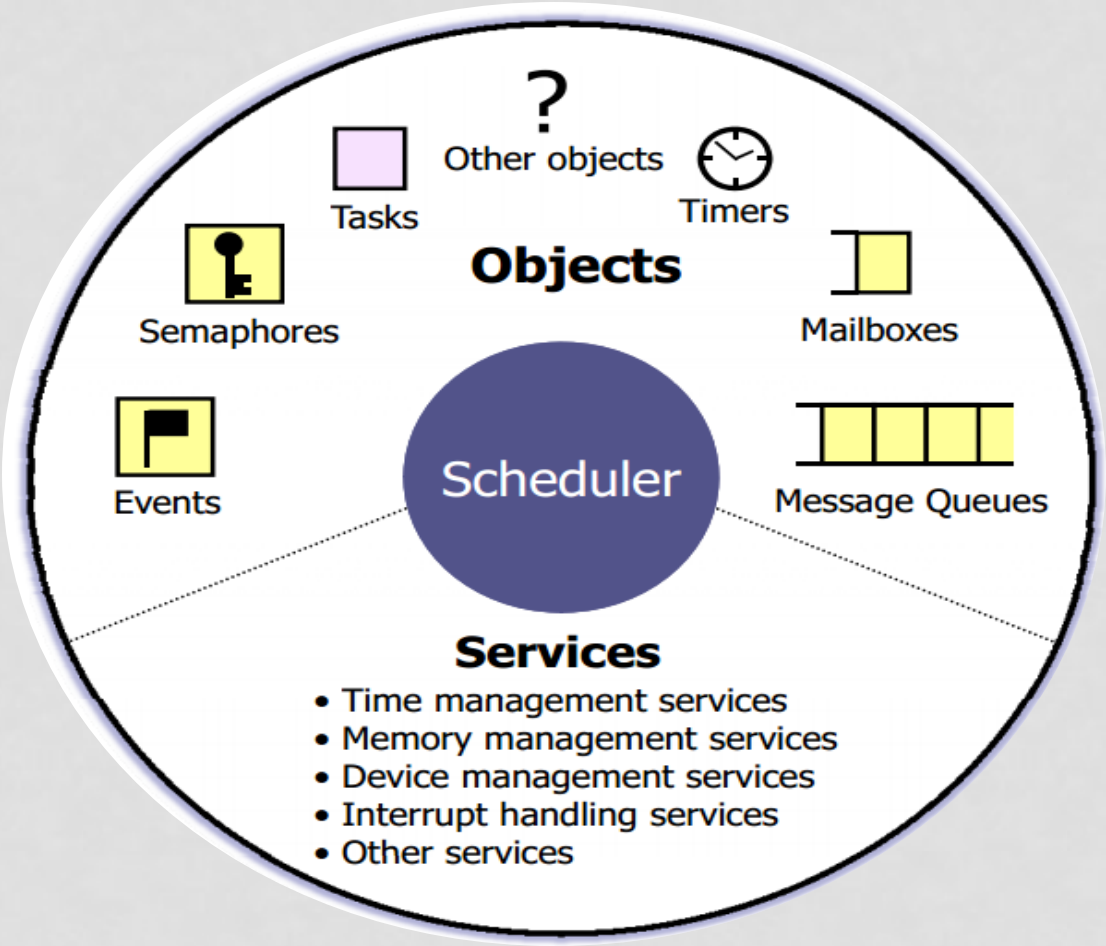
Kernel

- Most fundamental part of an OS upon which all other components rely.
- Responsible for the management of tasks and communication between tasks.
- In RTES,
  - RTOS may comprise only a RT kernel.
  - Combination of various modules including the Real Time Kernel.
- Use of RT Kernel simplifies the design of Embedded System.

REAL TIME KERNEL

Essential Components of Real Time Kernel:

- Scheduler
- Objects
- Services



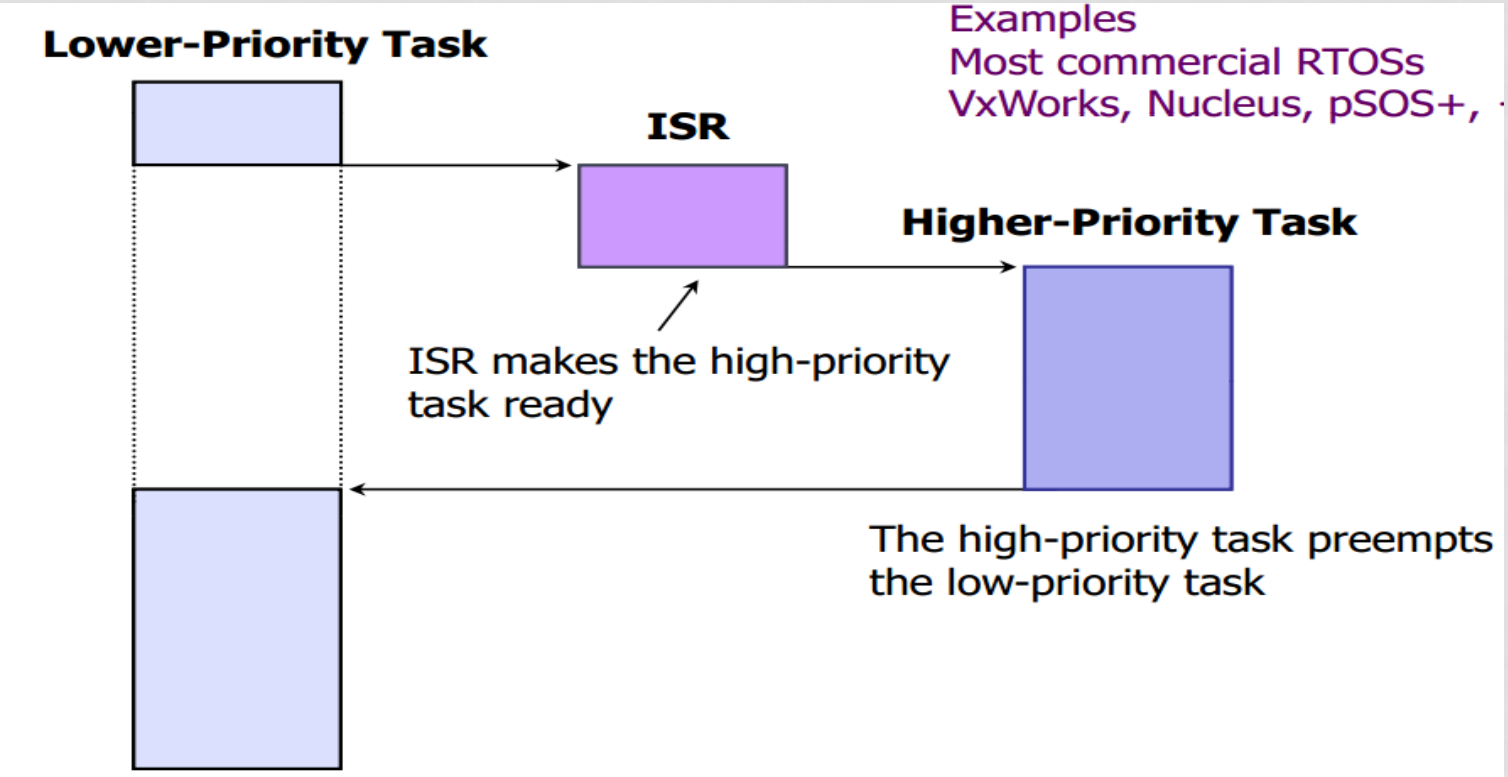
REAL TIME KERNEL

Scheduler:

- Heart of Kernel.
- Scheduler is an RTOS is simpleminded.
- Runs the highest priority task that is ready to run.
- Scheduler in Desktop OS is fair.
  - gives all task equal chance of execution.
- 2- scheduling policies:
  - Preemptive Scheduling.
  - Non-Preemptive Scheduling.

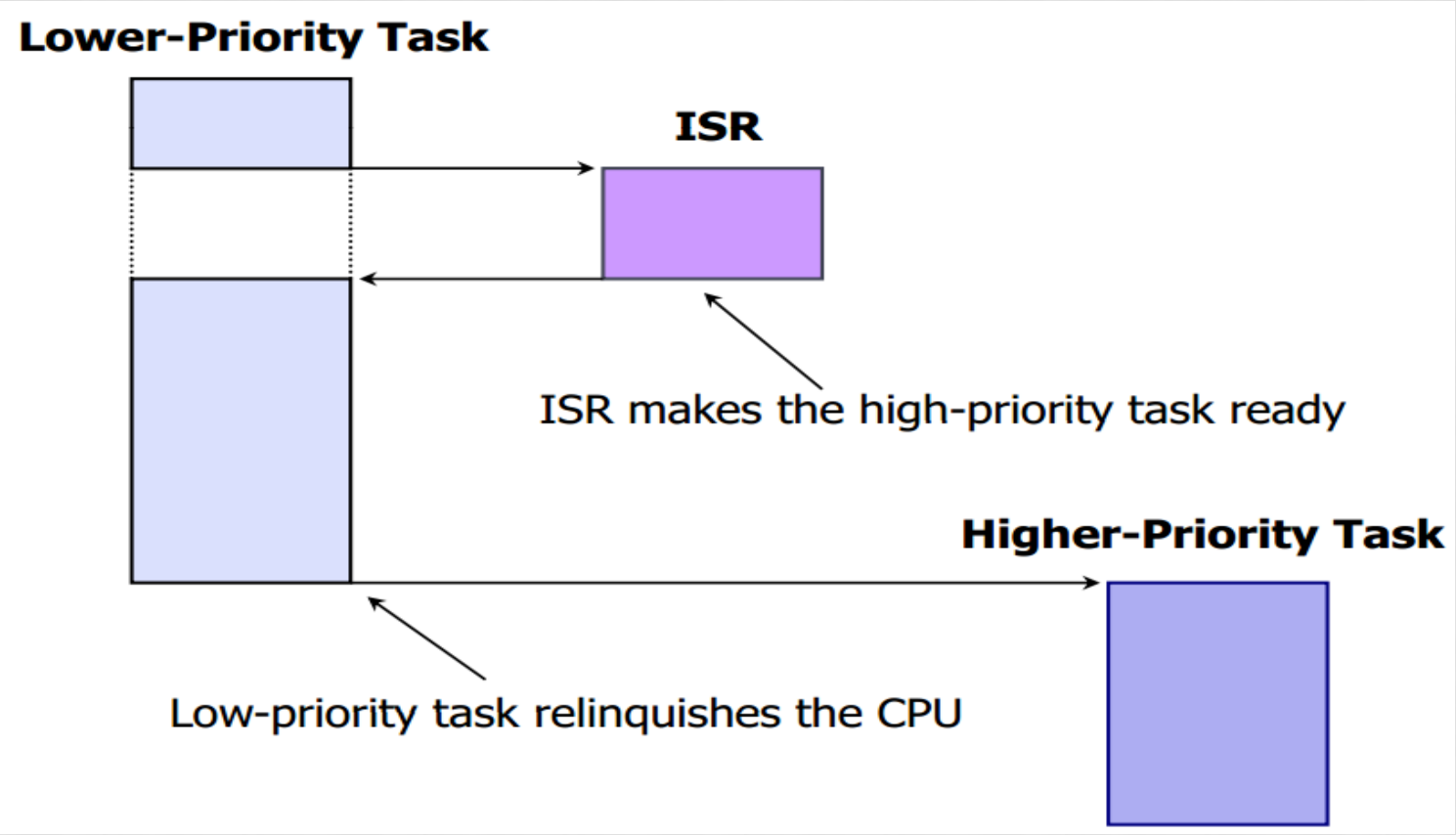
SCHEDULING

Preemptive Scheduling:



SCHEDULING

Non preemptive Scheduling:



SCHEDULING

Preemptive	Non-Preemptive
The resources are allocated to a process for a limited time.	Once resources are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Process can be interrupted in between.	Process can not be interrupted till it terminates or switches to waiting state.
If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
Preemptive scheduling has overheads of scheduling the processes.	Non-preemptive scheduling does not have overheads.
Preemptive scheduling is flexible.	Non-preemptive scheduling is rigid.