Introduction to Embedded Systems

Sem IV Unit V

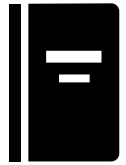
Mr. Umesh Koyande & Mr. Kiran Datar

Operating Systems Basics

Process

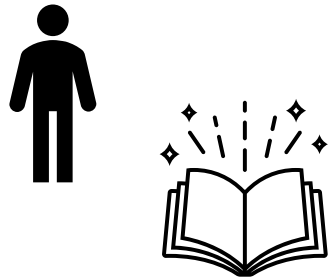
Person

Book



Passive

Process



Active

Process

CPU

Passive

Program

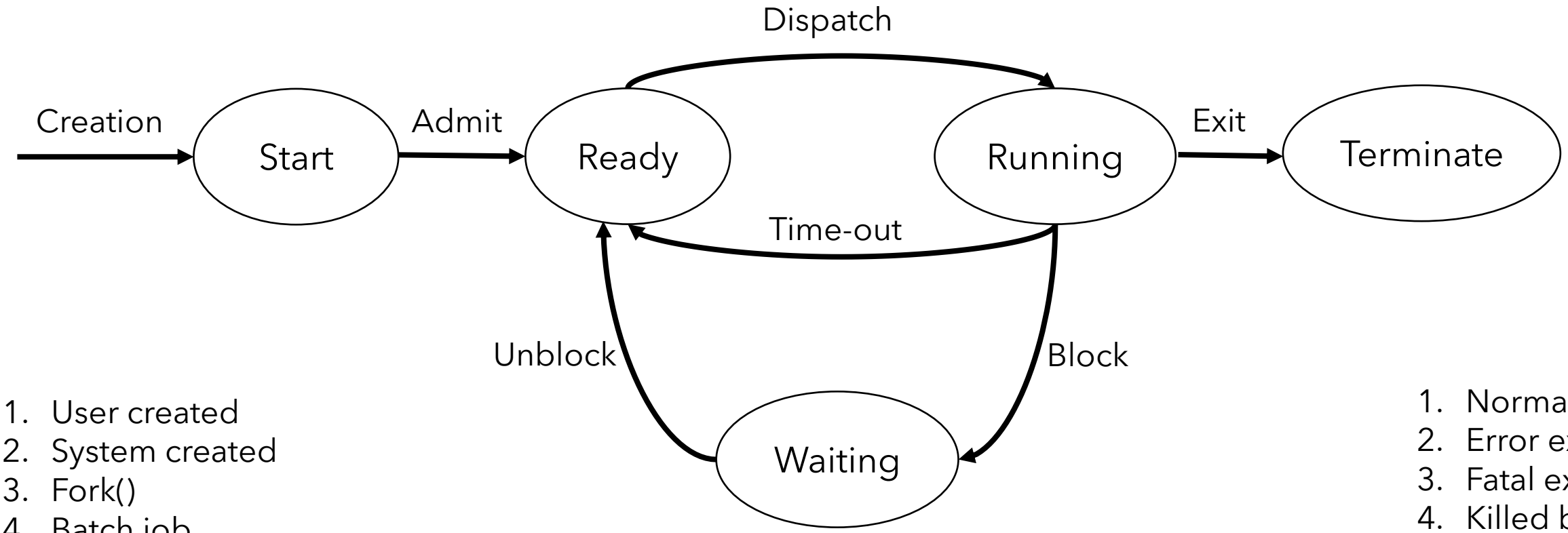
CPU

Active

Process

Process is a program in execution.

Process Life Cycle

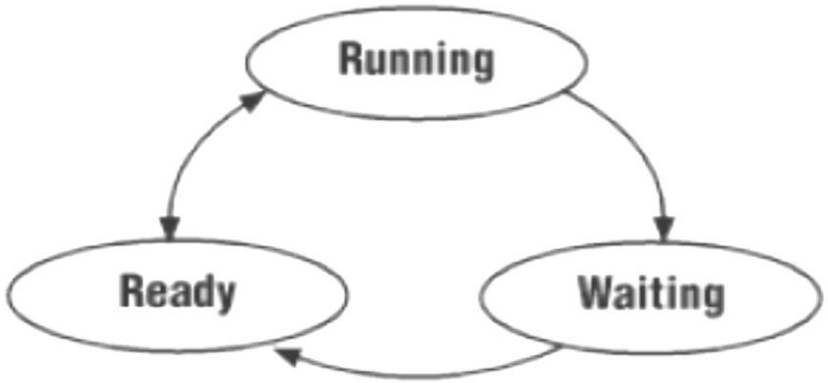


1. User created
2. System created
3. Fork()
4. Batch job

1. for resources
2. for input from user
3. Interrupt by high priority process

1. Normal Exit
2. Error exit
3. Fatal exit
4. Killed by another process

Task States



Ready-

- Ready to Run, not currently using the processor
- Waiting for CPU.

Running-

- Task in execution.
- No other task can be in that same state at the same time

Waiting-

- Waiting for resources- I/O devices
- waiting for some event external
- Interrupts- High priority process executed by CPU first

Scheduler

First-in-first-out (FIFO)

scheduling describes an operating system which is not a multitasking operating system at all. Rather, each task runs until it is finished, and only after that is the next task started

Repsly

Scheduler

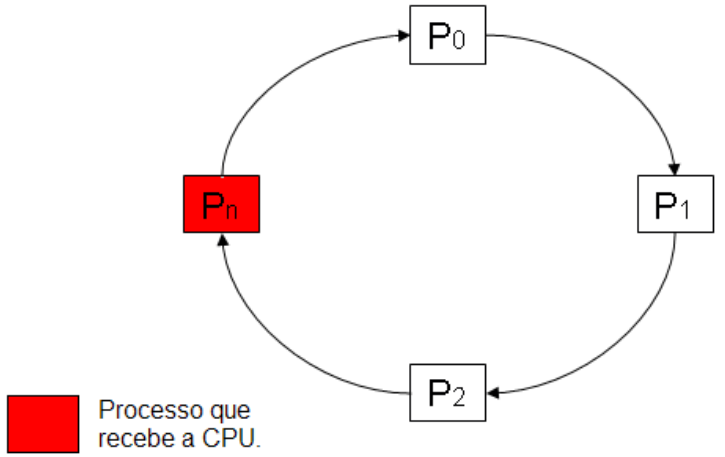
Shortest job first (SJF)

- Next task selected is the one that will require the least amount of processor time to complete.
- It has the appealing property of maximizing the number of satisfied customers.

Scheduler

Round robin (RR)

- **E**ach task runs for some predetermined amount of time.
- After that time interval has elapsed, the running task is preempted by the operating system and the next task in line gets its chance to run.
- The preempted task doesn't get to run again until all of the other tasks have had their chances in that round.



Scheduler

Priority Scheduling

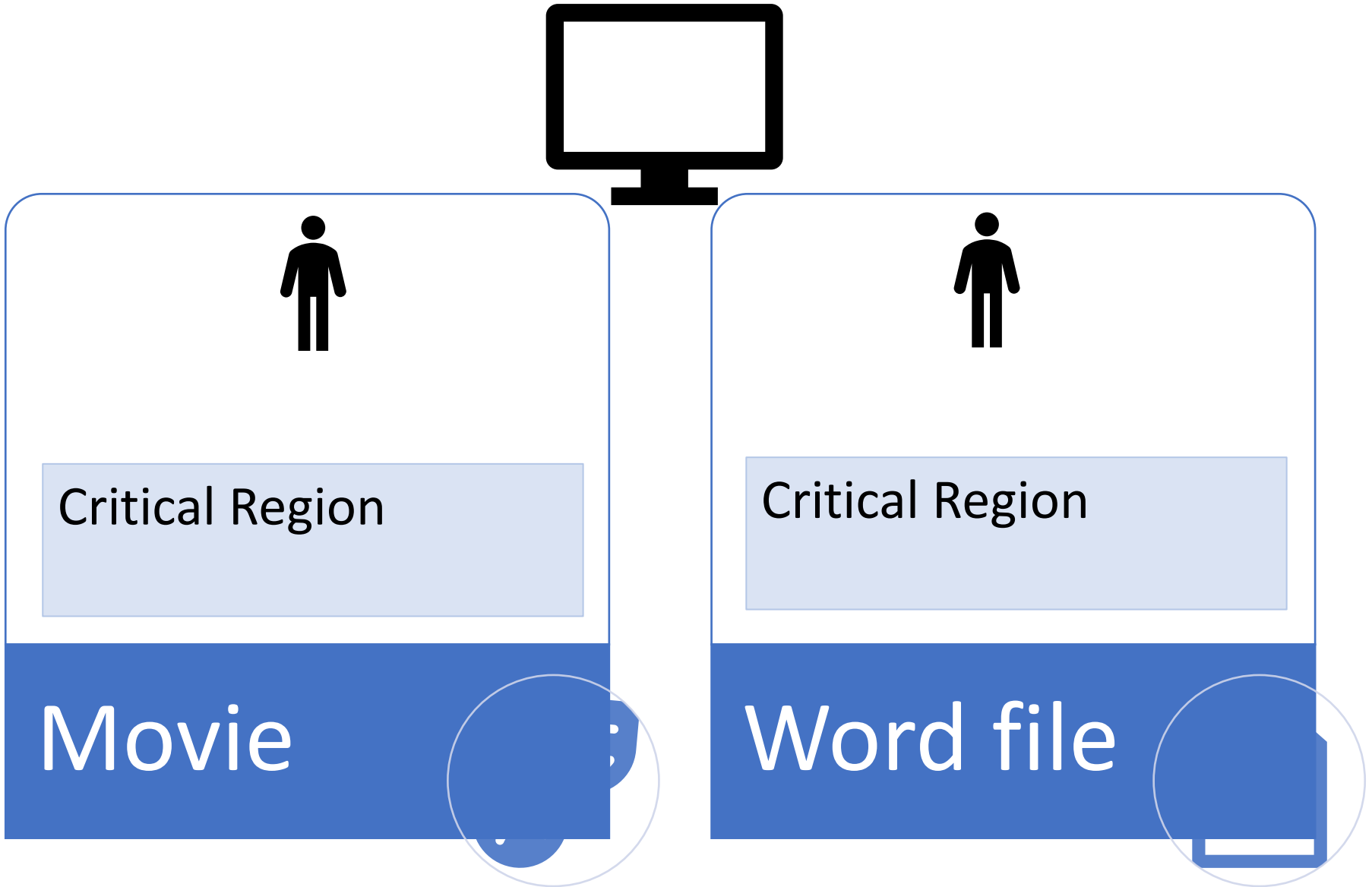
- Most embedded operating systems utilize a priority-based scheduling algorithm that supports preemption.
- Task that is currently using the processor is guaranteed to be the highest-priority task.
- Lower-priority tasks must wait until higher-priority tasks are finished using the processor before resuming their work.

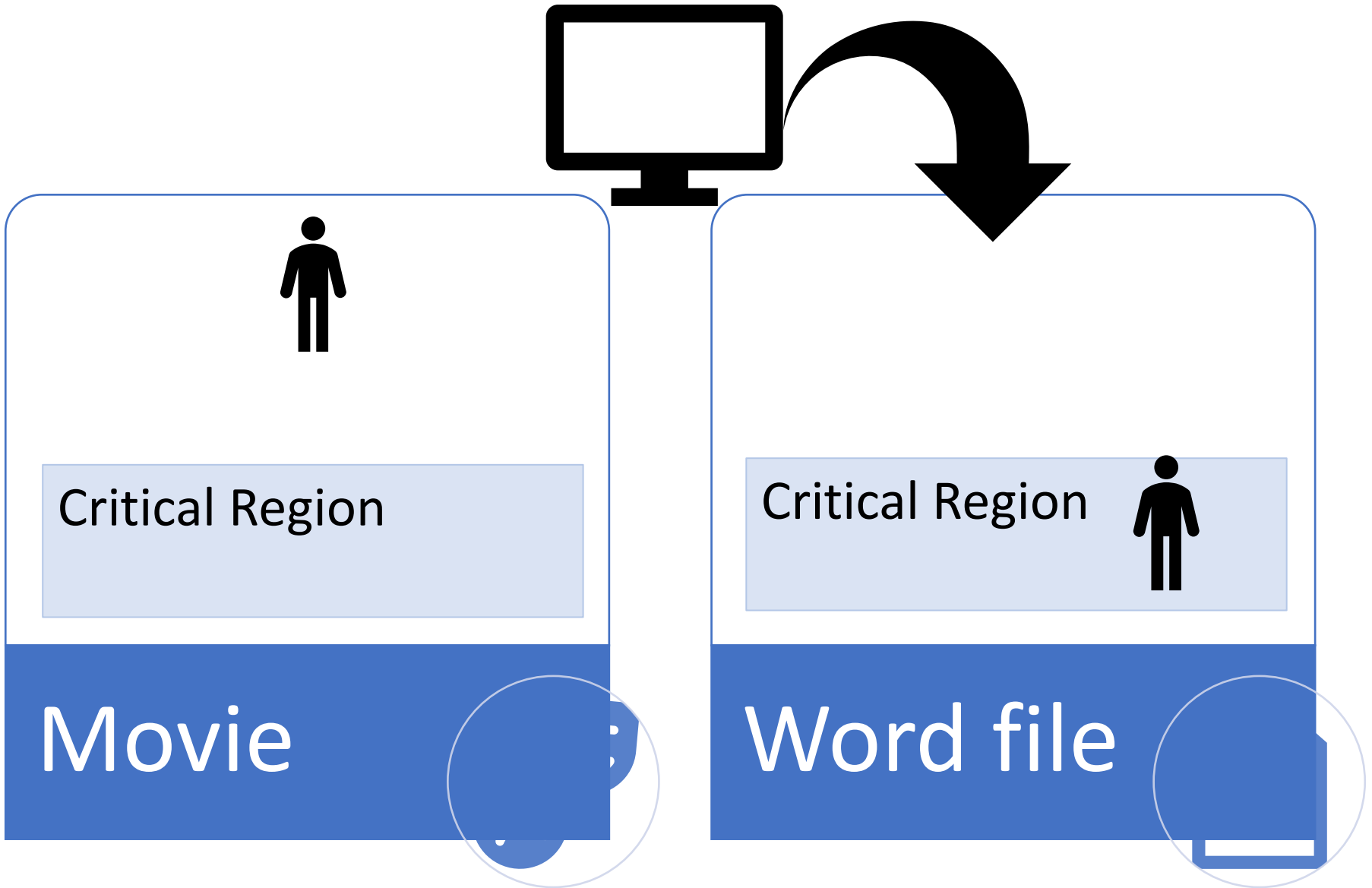
Task Synchronization

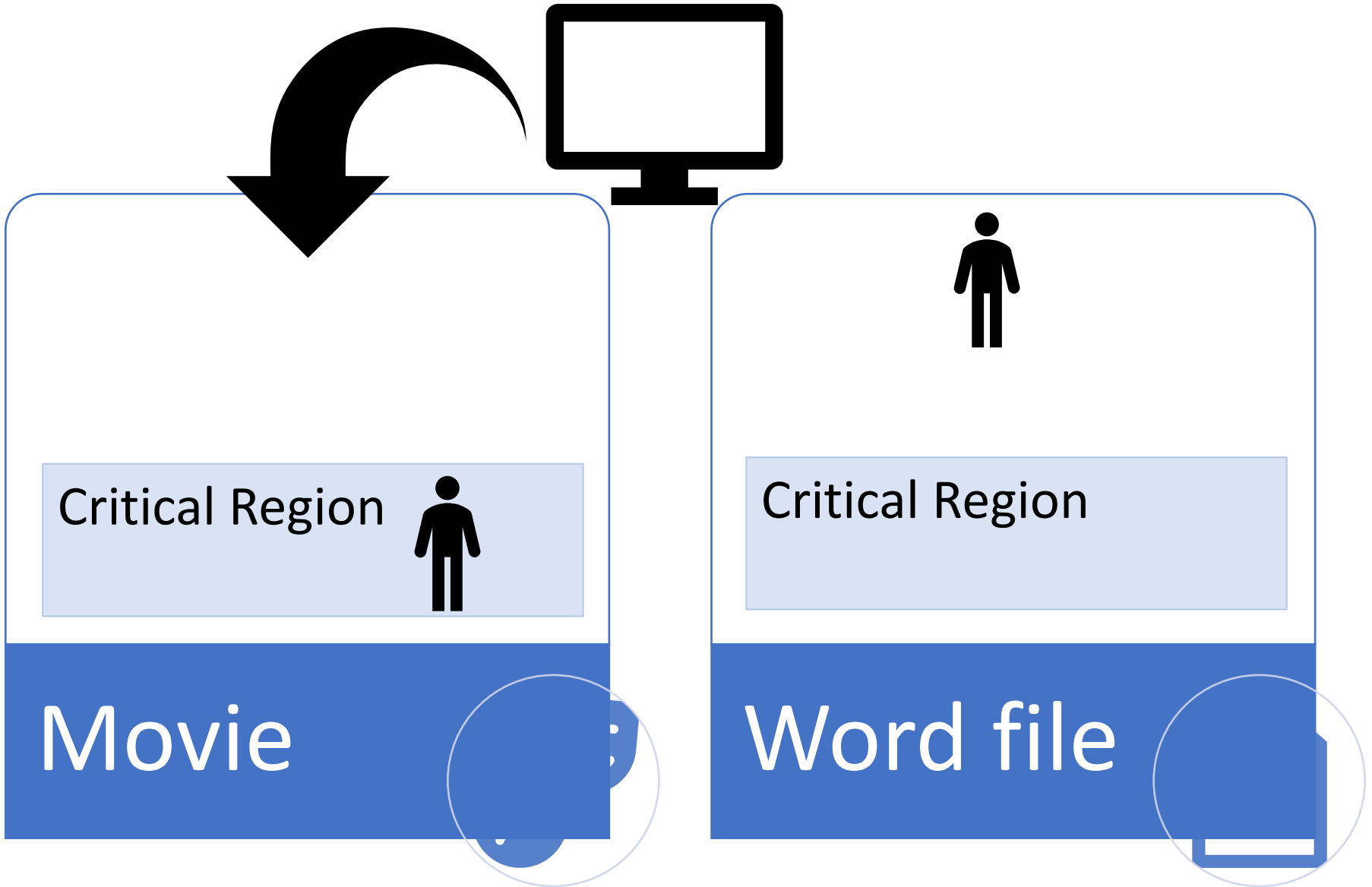
All of the tasks are working together to solve a larger problem and must occasionally communicate with one another to synchronize their activities

Mutexes are multitasking-aware because the processes of setting and clearing the binary flag are automatic

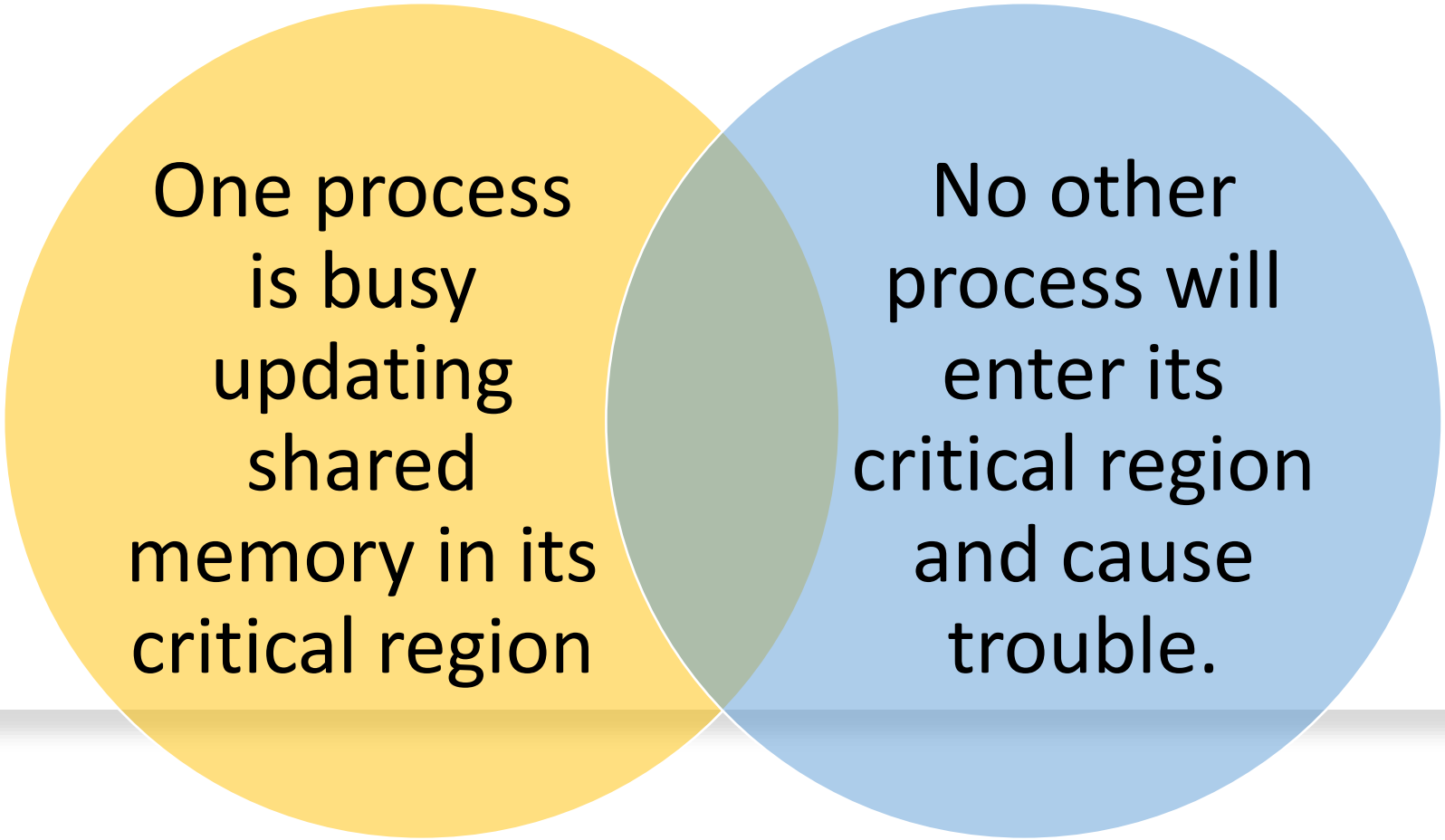
When this binary flag is set, the shared device is assumed to be in use by one of the tasks. All other tasks must wait until that flag is cleared


Critical Region
Movie
Critical Region
Word file


Critical Region
Movie
Critical Region
Word file


Critical Region
Movie
Critical Region
Word file

Mutual Exclusion



Mutual Exclusion

Processes want to access the shared data

Shared Resource like
Files, Printer, Memory

One process is
accessing shared data

Mutual Exclusion

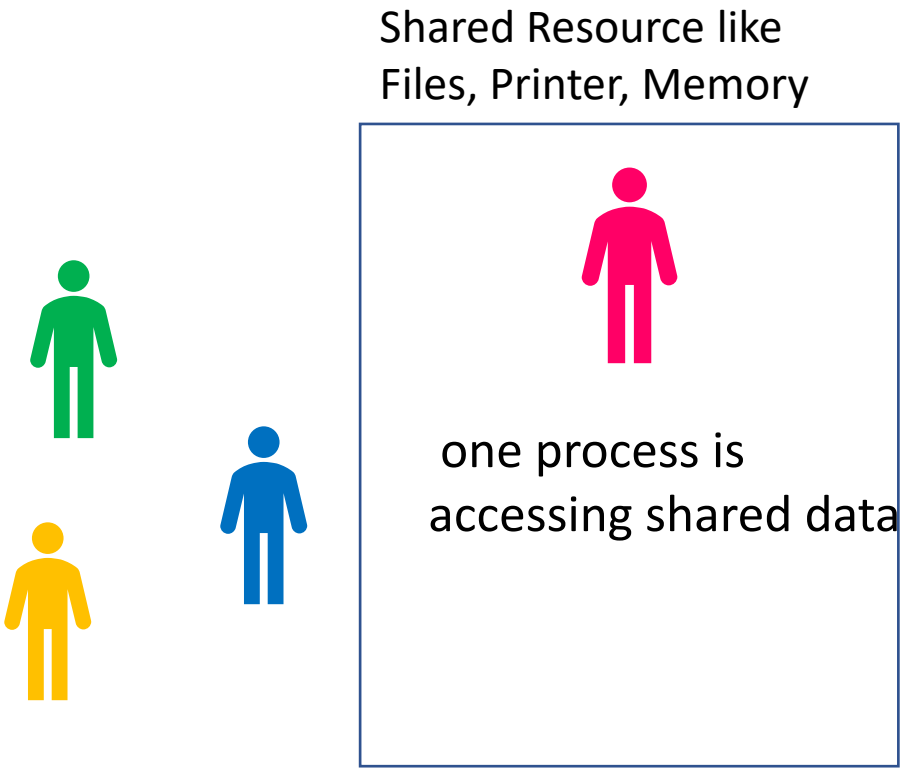
Shared Resource like
Files, Printer, Memory

But already one process is accessing the shared data, therefore others must

WAIT

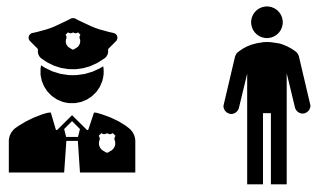
Mutual Exclusion and Busy Waiting

BUSY WAITING



Semaphore

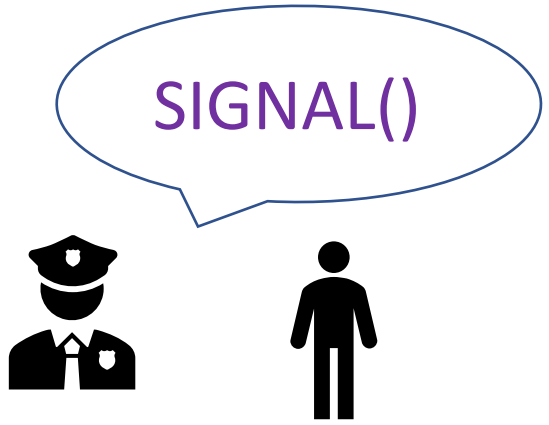
Semaphore



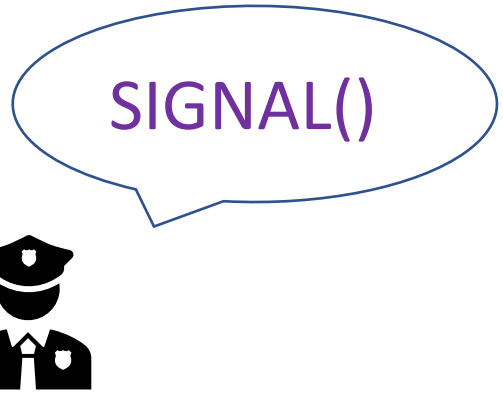
Semaphore

Semaphore

Semaphore



Semaphore



Semaphore

- A **semaphore** S is an integer variable
- Atomic operations: wait() and signal().

The definition of wait() is as follows:

```
wait(S)
{
while (S <= 0)
; // busy wait
S--;
}
```

The definition of signal() is as follows:

```
signal(S)
{
S++;
}
```

Mutex

A mutex is a shared variable that can be in one of two states: unlocked or locked.

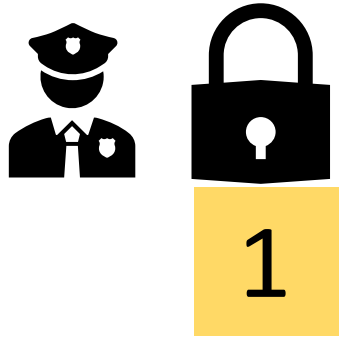
When the semaphore's ability to count is not needed, a simplified version of the semaphore, called a mutex, is used.

Mutexes are good only for managing mutual exclusion to some shared resource or piece of code.

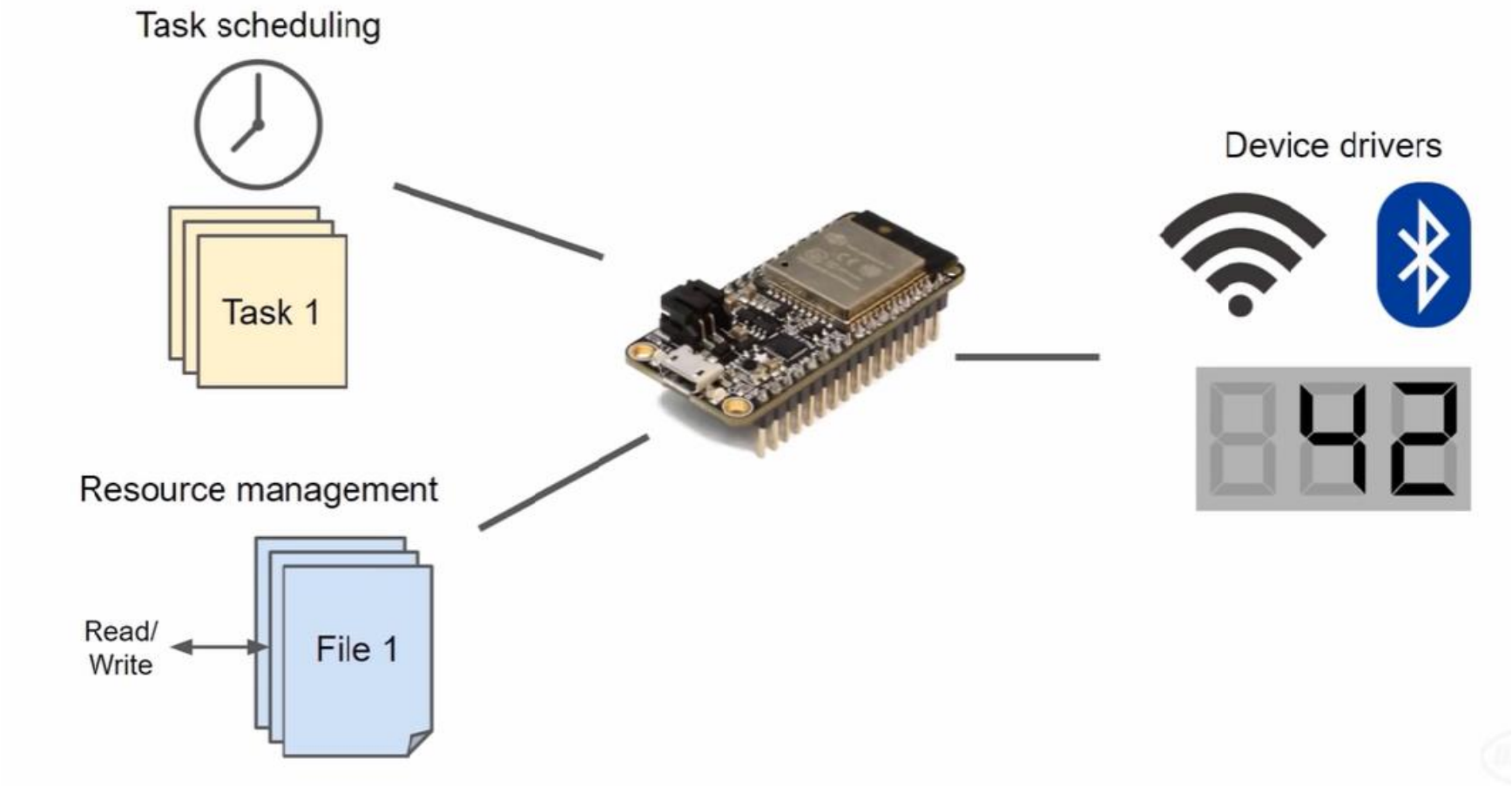
Only1 bit is required to represent it, with 0 meaning unlocked and all other values meaning locked

Mutex

Mutex



RTOS


Task scheduling
Task 1
Device drivers
42
Resource management
Read/
Write
File 1

RTOS

Real Time	To serve real time application that process data as it comes in, mostly without buffer delay
Time bound	Time-bound system that can be defined as fixed time constraints
Event driven	Event-driven with no time wastage on processing time for the event which is not occur
Memory	Occupy very less memory
Resource	Consume fewer resources

RTOS

Hard Real Time :

- In Hard RTOS, the deadline is handled very strictly which means that given task must start executing on specified scheduled time, and must be completed within the assigned time duration.
- Example: Medical critical care system, Aircraft systems, etc.

Soft Real Time:

- Soft Real time RTOS, accepts some delays by the Operating system. In this type of RTOS, there is a deadline assigned for a specific job, but a delay for a small amount of time is acceptable. So, deadlines are handled softly by this type of RTOS.
- Example: Online Transaction system and Livestock price quotation System.

Real-Time Characteristics

- An operating system is said to be **deterministic** if the worst-case execution time of each of the system calls is calculable
- RTOS seriously will usually publish a data sheet that provides the minimum, average, and maximum number of clock cycles required by each system call
- These numbers might be different for different processors, but it is reasonable to expect that if the algorithm is deterministic on one processor, it will be so on any other.

Real-Time Characteristics

Interrupt latency is the total length of time from an interrupt signal's arrival at the processor to the start of the associated interrupt service routine.

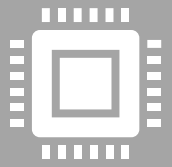
When an interrupt occurs, the processor must take several steps before executing the ISR.

- Processor must finish executing the current instruction.
- Interrupt type must be recognized.
- Only if interrupts are enabled, the ISR that is associated with the interrupt is started.

If interrupts are ever disabled within the operating system, the worst-case interrupt latency increases by the maximum amount of time that they are turned off

Real-Time Characteristics

Amount of time required to perform a **context switch**



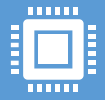
There is no magic number and the actual times are usually processor-specific because they are dependent on the number of registers that must be saved and where.

Disassembler/ Decompliler

Disassembler/Decompliler is a reverse engineering tool.

Reverse engineering is the process of revealing the technology behind the working of a product.

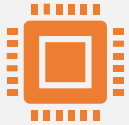
Disassemblers/decompilers help the reverse-engineering process by translating the embedded firmware into Assembly/high level language instructions.



Disassembler: It is a program which converts machine codes into assembly code (instructions)

Decompiler: It is the program which converts machine codes into corresponding high level language

Disassembler/Decompliler



Disassemblers/Decompilers are powerful tools for analyzing the presence of malicious codes (virus information) in an executable image.

It is not possible for a disassembler/decompiler to generate an exact replica of the original assembly code/high level source code in terms of the symbolic constants and comments used.

However disassemblers/decompilers generate a source code which is somewhat matching to the original source code from which the binary code is generated.

Simulator

Software tool used for simulating the various conditions for checking the functionality of the application firmware.

It doesn't require a real target system

Simulator based debugging technique is simple and straight forward.

Very primitive (Lack of featured I/O support Everything is a simulated one)

Do not show real-time behavior

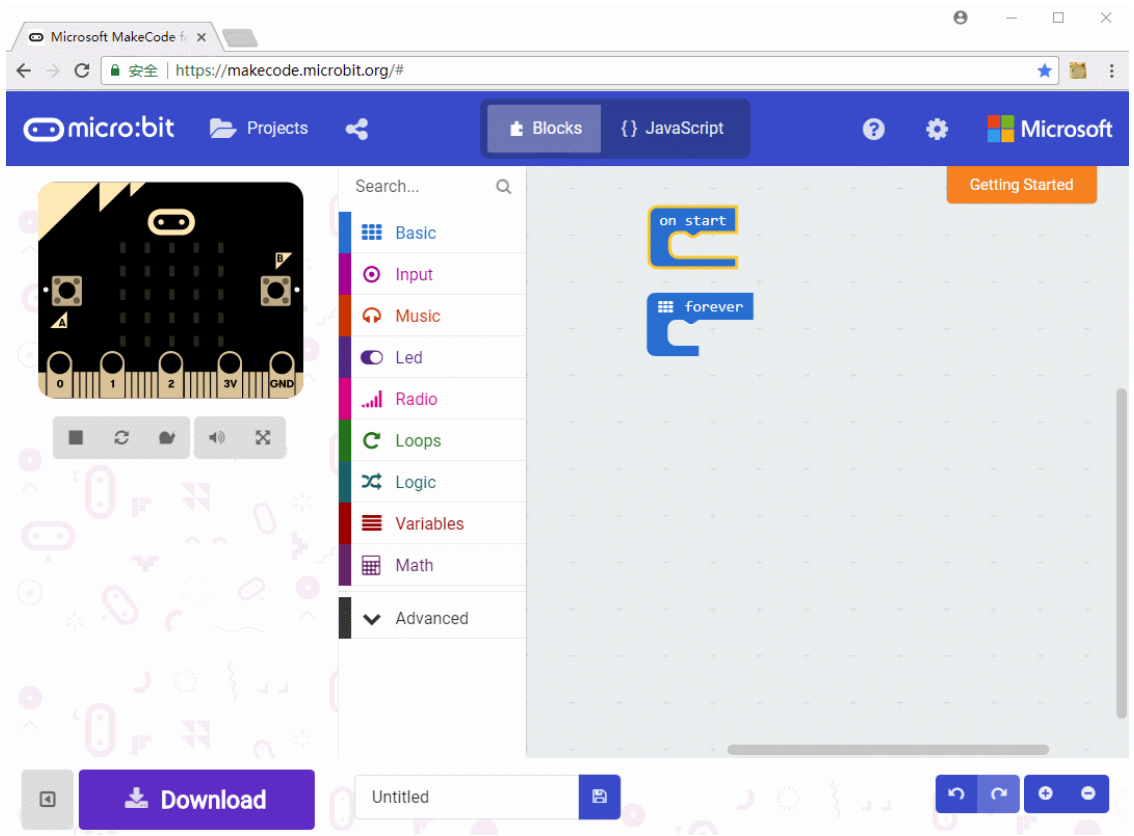
Debugging is developer driven and it is no way capable of creating a real time behaviour.

Advantages of simulator based debugging

No need for original target board

- Purely software oriented.
- Since the real hardware is not required, firmware development can start well in advance
- Saves development time.

Advantages of simulator based debugging



Simulate I/O peripherals

- Simulator provides the option to simulate various I/O peripherals.
- Using simulator's I/O support one can edit the values for I/O registers
- Hence it eliminates the need for connecting I/O devices

Advantages of simulator based debugging

Simulates abnormal conditions

- One can input any desired value for any parameter during debugging the firmware and can observe the control flow of firmware.
- It helps firmware developer to study the behavior of firmware under abnormal input conditions

Limitations of simulator based debugging

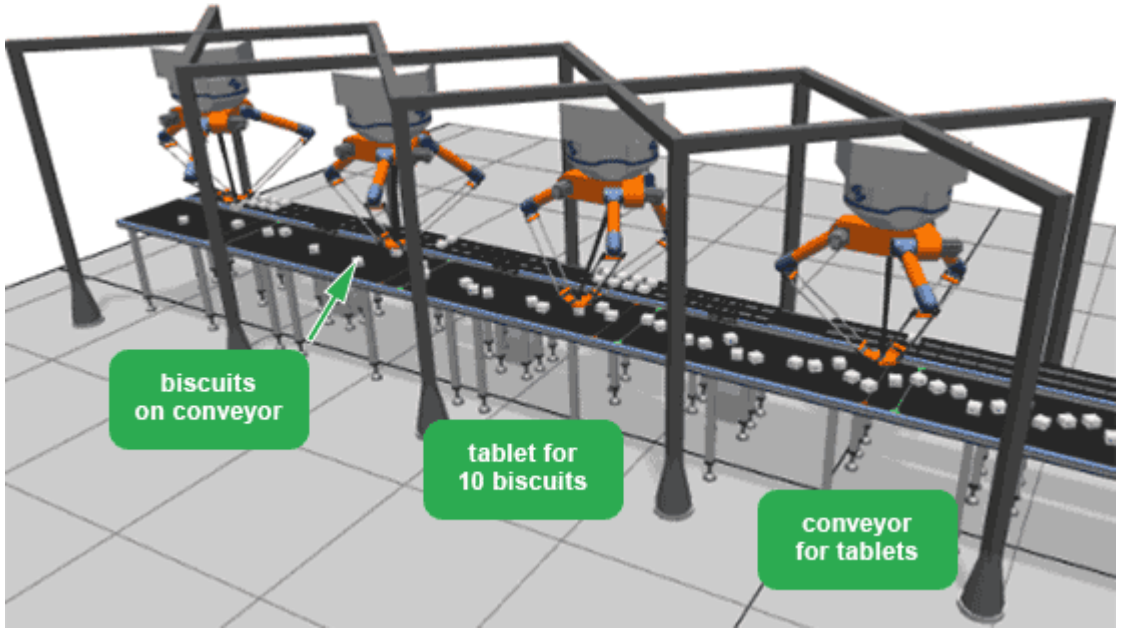
Deviations from real behavior

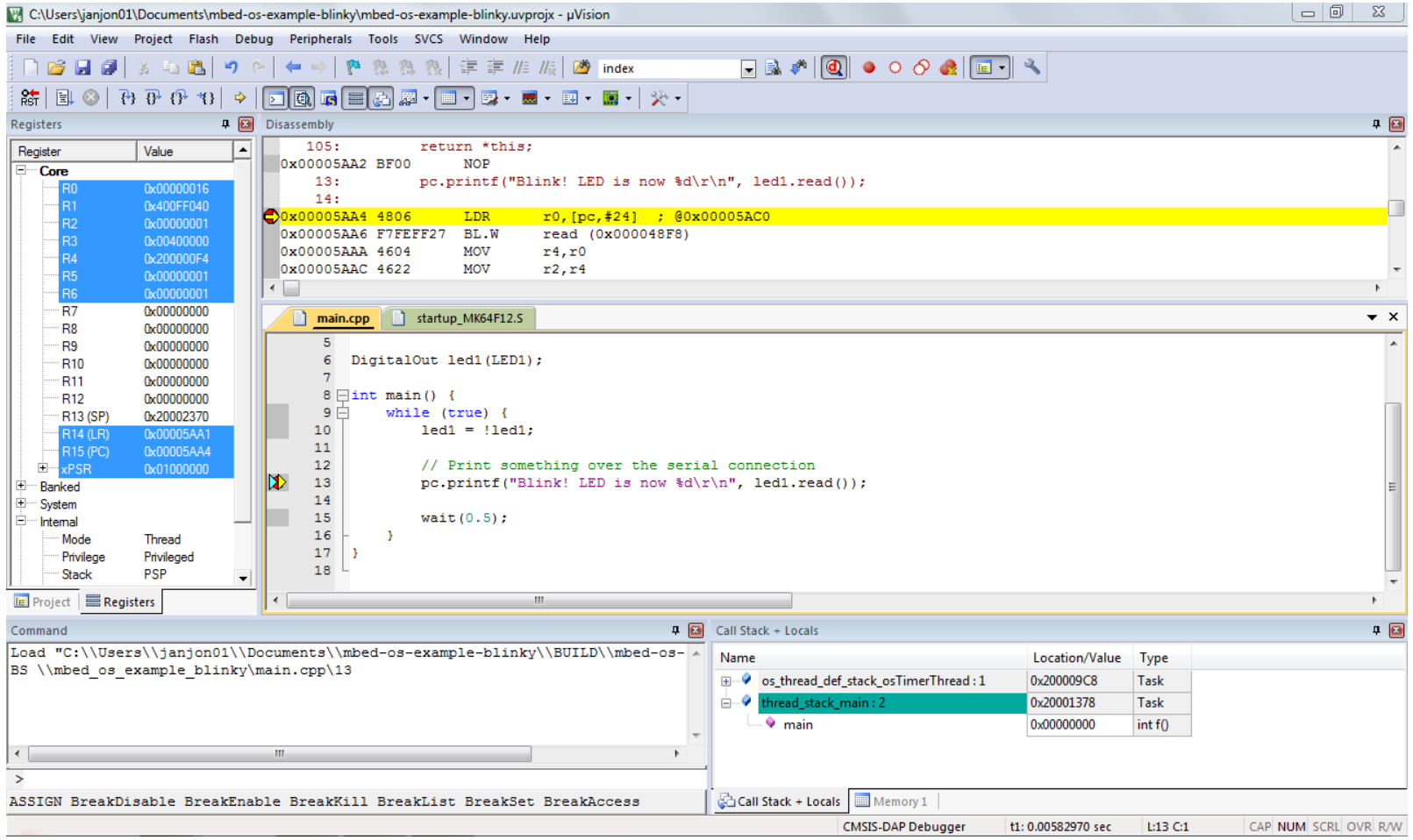
- Since developer is not using actual hardware, therefore he may not be able to debug all possible conditions.
- Under certain operating conditions we may get some particular result and it need not be the same when the firmware runs in a production environment.

Limitations of simulator based debugging

Lack of real timeliness

- Do not show real-time behavior
- In a real application the I/O condition may be varying or unpredictable





Integrated Development Environment (IDE)

https://www.youtube.com/watch?v=oBvP_Lxoqxo&ab_channel=MicrotronicsTechnologies

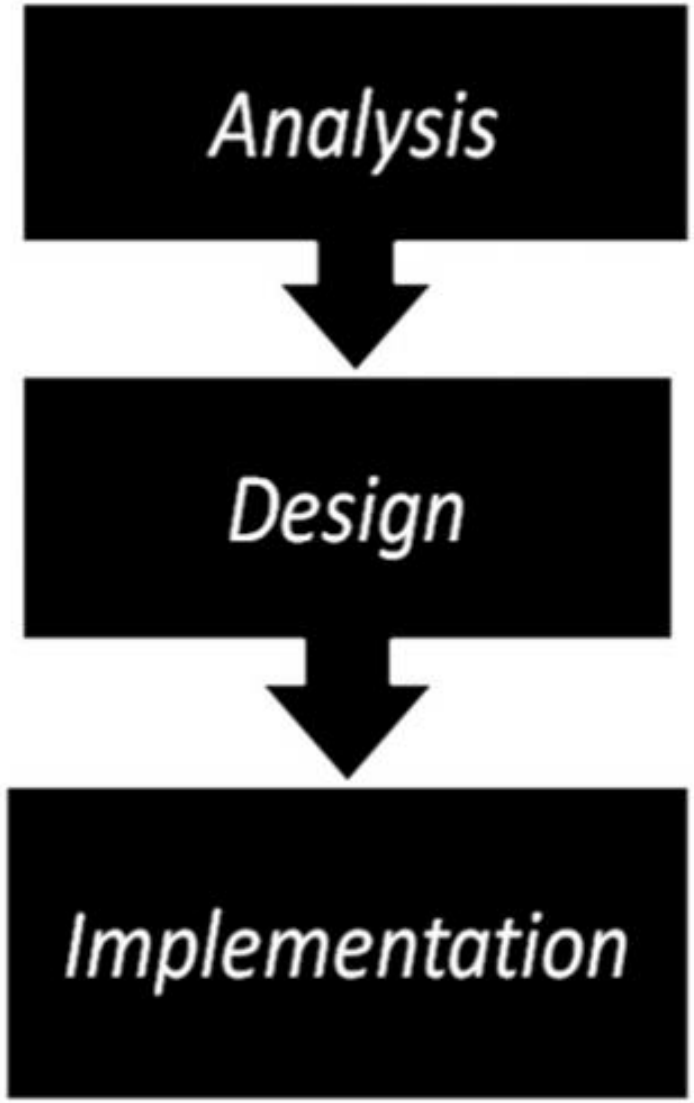
Types of files generated on cross compilation

- *List File (.lst)*
- *Hex File (.hex)*
- *Pre-processor Output.file,*
- *Map File (File extension linker dependent)*
- *Object File (.obj)*

EDLC

Embedded-product Development Life Cycle

- Based on 'Analysis-Design-Implementation' approach for embedded-product development
- Ultimate aim of any embedded product in a commercial production setup is to produce Marginal benefit.
- Return On Investment
- Investment for product development includes initial investment, manpower, infrastructure investment etc.



Objectives of EDLC

Ensure that high quality products are delivered to user

- Quality in any product development is Return On Investment achieved by the product
- Expenses incurred for developing the product are:-
  - Initial investment
  - Developer recruiting
  - Training
  - Infrastructure requirement related

Objectives of EDLC

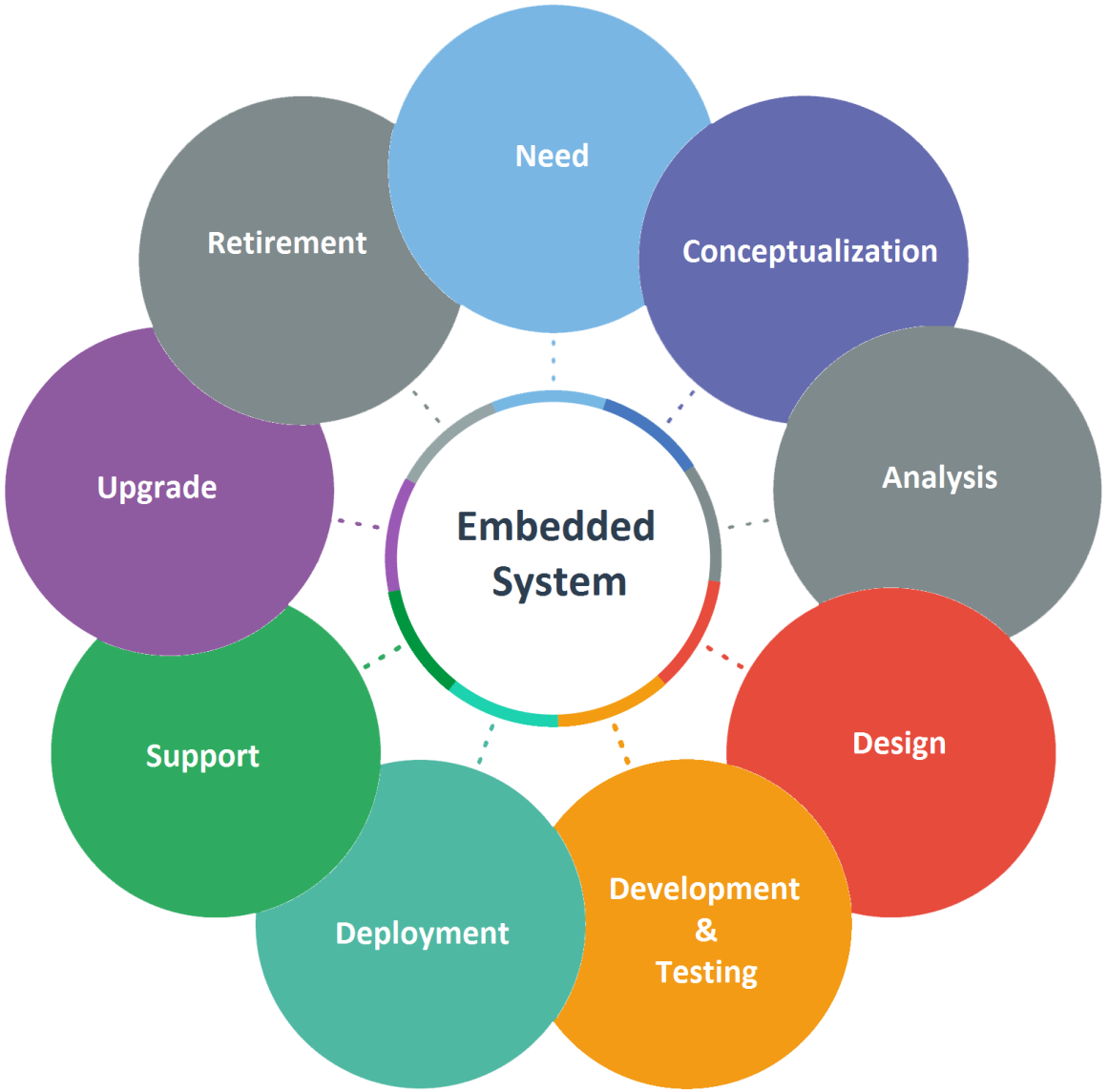
Risk minimization defect prevention in product development

- Product development may require ‘loose’ or ‘tight’ project management
- Project management is essential for predictability co-ordination and risk minimization
- Resource allocation is critical and it is having a direct impact on investment

Objectives of EDLC

Maximize the productivity

- Productivity is a measure of efficiency as well as Return On Investment
- Based on total manpower efficiency
- Productivity is said to be increased, if the product is capable of yielding maximum returns with reduced investment.
- Saving manpower effort will definitely result in increased productivity



Phases of EDLC

Need

Based on the need for the product, a 'Statement of Need' or 'Concept Proposal' is prepared

New or Custom Product Development:

- Need for a product which does not exist in the market

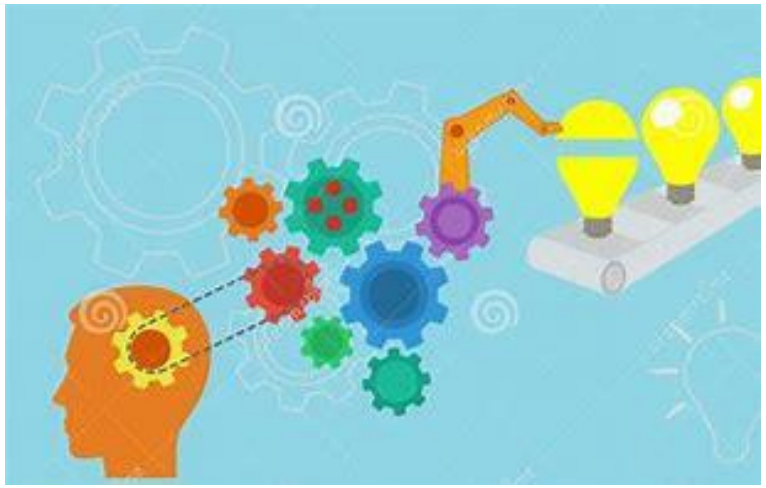
Product Re-engineering:

- In order to sustain in the market, constantly improve an existing product by incorporating new technologies and design

Product Maintenance:

- Product maintenance 'need' deals with providing technical support to the end user for an existing product in the market

Conceptualization



Defines the scope of the concept, performs cost benefit analysis and feasibility study and prepares project management and risk management plans.

Feasibility Study

- Technical and Financial feasibility

Cost Benefit Analysis (CBA)

- Revealing and assessing the total development cost and profit expected from the product.

Product Scope

- Deals with the activities involved in the product to be made.

Planning Activities

- Requires various plans to be developed first before development like Resource Planning & Risk management Plans.

Analysis

Analysis and Documentations

Requirements that need to be addressed

Defining Test Plan and Procedures

- Unit testing
- Integration testing
- System testing
- User acceptance testing

Design

Identifies application environment and creates an overall architecture for the product.

Starts with the Preliminary Design and establishes the top level architecture for the product.

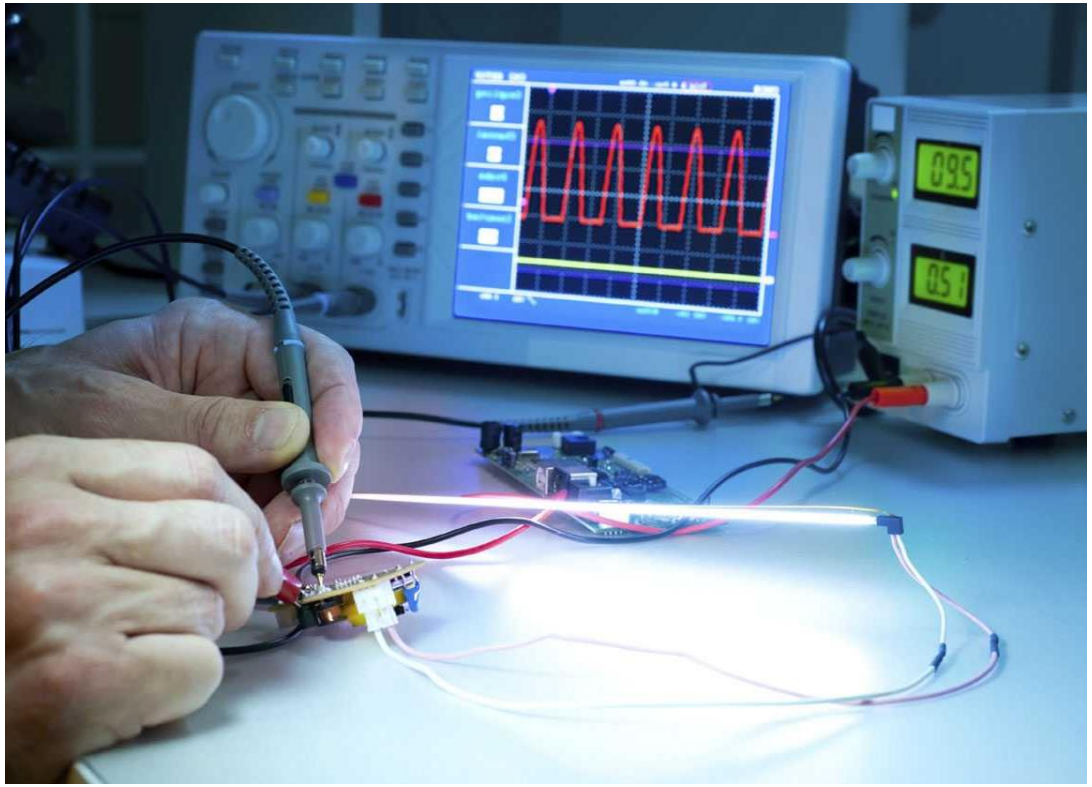
On completion it resembles a ‘black box’ that defines only the inputs and outputs.

Final product is called Preliminary Design Document (PDD).

Once the PDD is accepted by the End User the next task is to create the ‘Detailed Design’.

It encompasses the Operations manual design, Maintenance Manual Design and Product Training material Design and is together called the ‘Detailed Design Document’.

Development and Testing



Development phase transforms the design into a realizable product.

The detailed specification generated during the design phase is translated into hardware and firmware.

The Testing phase can be divided into independent testing of firmware and hardware that is:

- Unit testing
- Integration testing
- System testing
- User acceptance testing

Deployment

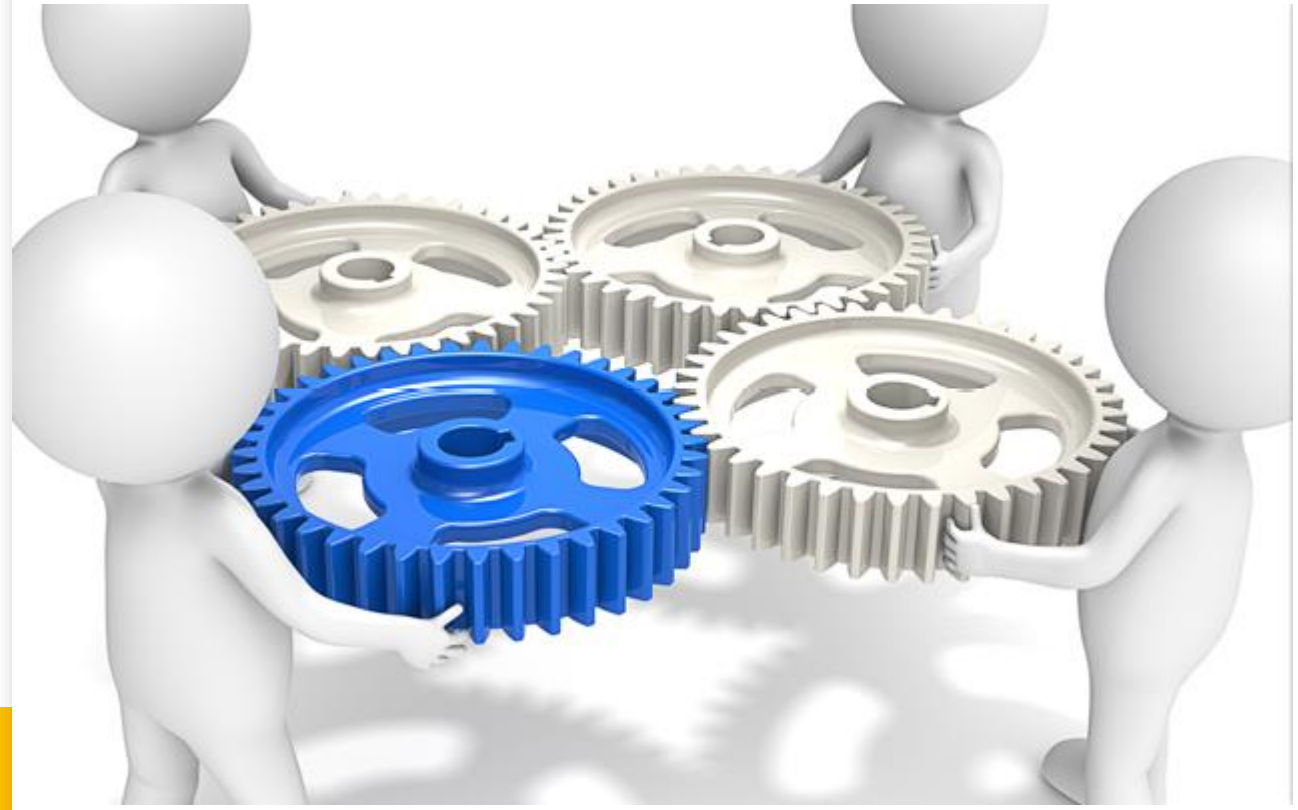
Deployment is the process of launching the first fully functional model of the product in the market.

It is also known as First Customer Shipping (FCS).

Tasks performed during this phase are:

- **Notification of Product Deployment**
  - Deployment schedule
  - Brief description about the product
  - Targeted end user
  - Extra features supported
  - Product support information
- **Execution of training plan**
  - Proper training should be given to the end user top get them acquainted with the new product.
- **Product installation**
  - Install the product as per the installation document to ensure that it is fully functional.
- **Product post Implementation Review**
  - After the product launch, a post implementation review is done to test the success of the product.

Support



- The support phase deals with the operational and maintenance of the product in the production environment.
- Bugs in the product may be observed and reported.
- The support phase ensures that the product meets the user needs and it continues functioning in the production environment.
- Activities involved under support are
  - **Setting up of a dedicated support wing**: Involves providing 24 x 7 supports for the product after it is launched.
  - **Identify Bugs and Areas of Improvement:** Identify bugs and take measures to eliminate them.

Upgrades

Deals with the development of upgrades (new versions) for the product which is already present in the market.

Product upgrade results as an output of major bug fixes.

During the upgrade phase the system is subject to design modification to fix the major bugs reported.

Retirement/Disposal

Retirement/disposal of the product is a gradual process.

This phase is the final phase in a product development life cycle where the product is declared as discontinued from the market.

The disposal of a product is essential due to the following reasons

- Rapid technology advancement
- Increased user needs


Thank
You!