Princípios da Computação

Building systems

Specific purpose systems

Specific-purpose system

- A specific-purpose system is a system designed to perform a particular set of functions within a defined environment, with limited or no interaction with external systems.

- It operates independently, often with tightly controlled inputs and outputs, and is optimised for reliability, security, and efficiency.

- Examples include embedded systems in medical devices, automotive control systems, and industrial automation, where strict performance and safety requirements are critical.

System applications without an OS

- Many embedded systems, IoT devices, real-time control systems, and dedicated industrial machinery do not make use of operating systems.

- Applications are designed and developed to run directly on hardware without an OS layer.

- Benefits include:

    - Lower resource usage (CPU, memory).

    - Faster startup and response times.

    - Precise control over hardware operations.

Development

- Code to interact directly with hardware registers and memory addresses (accessing GPIO ports, timers, or specific memory-mapped peripherals).

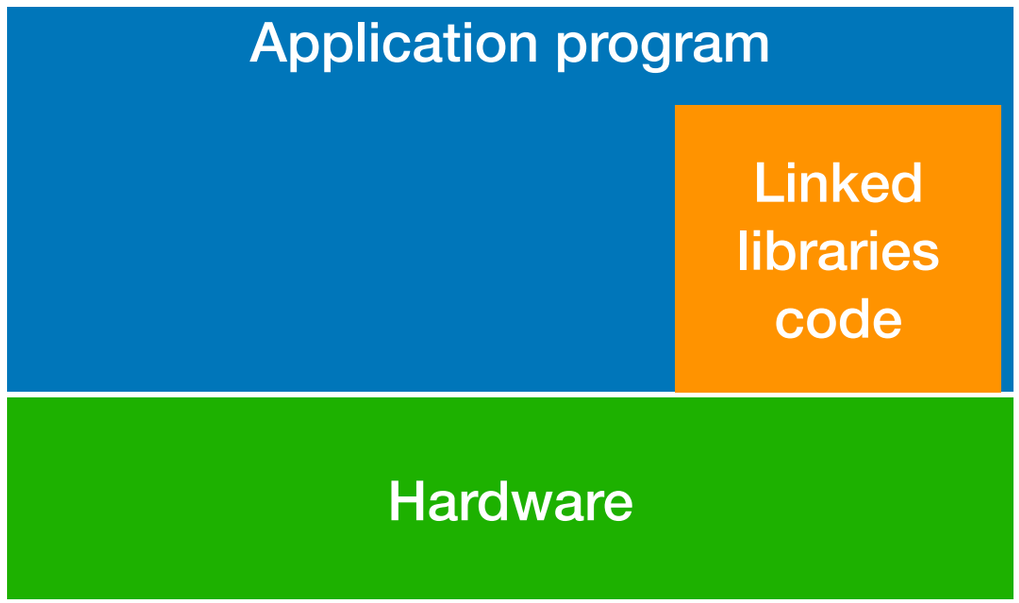
- Utilise vendor-provided libraries or SDKs that offer low-level functions for interacting with the hardware.

- Configure compiler and linker toolchains to ensure code is optimized and compatible with the hardware platform.

- Define memory layout, stack/heap size, and interrupt vectors specific to the hardware, directly in code.

- Perform debugging on the actual hardware due to tight integration with specific platform libraries.

Result


Application program
Linked libraries code
Hardware


flightradar24
ADS-B RECEIVER
Power
Mode-S
GPS
Extension

Real-Time Operating System (RTOS)

- A Real-Time Operating System (RTOS) is designed to handle time-sensitive applications by ensuring tasks are completed within specific timing constraints.
- Each task handles a distinct aspect of the system.
- Benefits include:
    - **Efficient Task Management.** Allows multi-tasking with minimal overhead.
    - **Reliability.** Ensures predictable performance for critical systems.
    - **Modular Development.** Enables easy integration of various components and subsystems.

What does an RTOS provide?

- **Inter-task Synchronisation.** Prevents tasks from interfering with each other during critical operations.

- **Inter-task Communication.** Controlled mechanisms for secure data exchange between tasks.

- **Memory Allocation.** Ensures sufficient memory availability throughout application runtime.

- **Deterministic Scheduling.** Guarantees tasks complete within defined time constraints.

- **Bounded Interrupt Handling.** Manages hardware interrupts to ensure timely execution of critical tasks.

RTOS application development process

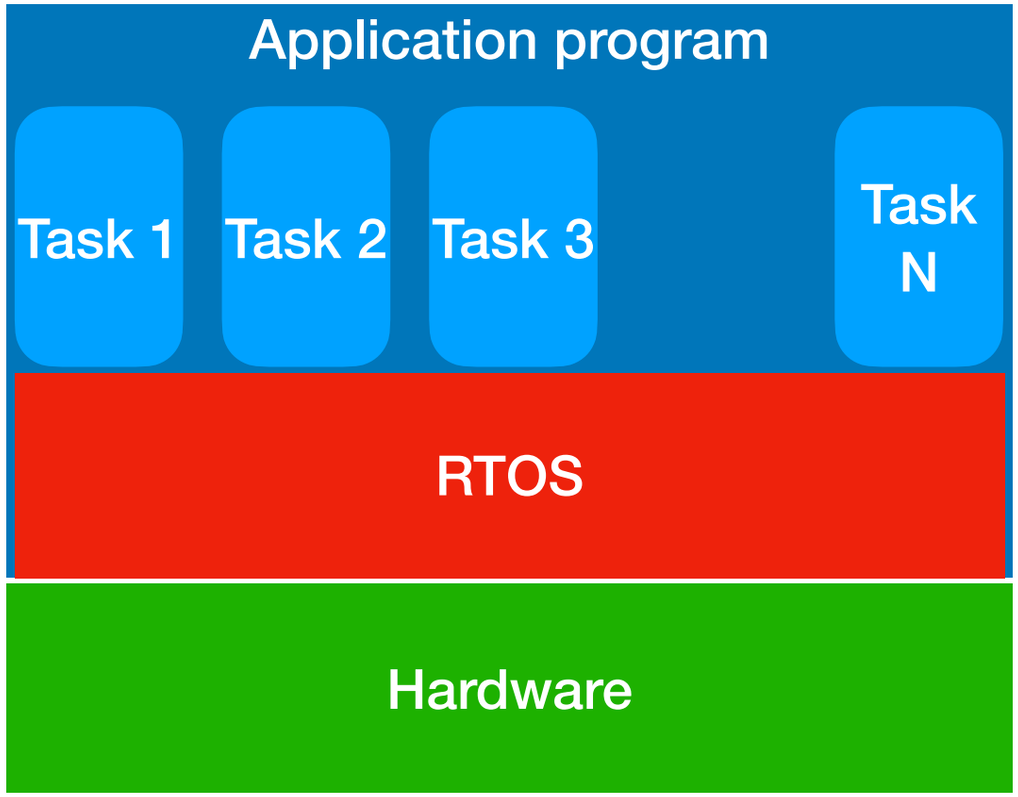
- Definition of set of concurrent tasks based on system requirements, each with specific priorities and deadlines.

- The RTOS provides system services like task scheduling, inter-task communication, and hardware abstraction. Application code calls these RTOS services through APIs for multitasking, synchronization, and timing management.

- The RTOS kernel, along with task management, timing, and resource handling, is compiled and linked together with the application’s tasks into a single executable. This executable is then loaded onto the target hardware.

- In most cases, tasks are structured as functions or threads managed by the RTOS, with their own stack and memory allocations. The RTOS executable maintains deterministic execution by prioritizing tasks according to the defined timing requirements, ensuring that real-time constraints are met.

Result


Application program
Task 1
Task 2
Task 3
Task N
RTOS
Hardware

General purpose systems

isep Instituto Superior de Engenharia do Porto

General purpose system

- A general-purpose system is a versatile computing system designed to perform a wide range of tasks across various applications.

- It is able to run different types of software to handle diverse user needs.

- Typically designed to interact with multiple peripherals and support various programs, making them adaptable for tasks like word processing, web browsing, data analysis, gaming, and more.

- Examples of general-purpose systems include personal computers, smartphones, and tablets, which provide flexibility, user interaction, and extensive functionality across different fields.

Challenges with general-purpose systems

- **Unpredictable Program Execution.** It's impossible to know in advance what programs will run on the system.

- **Dynamic Program Combination.** The combination of programs coexisting on the system during runtime cannot be predetermined.

- **Variable Resource Usage.** The system cannot anticipate how each program will utilise resources, leading to unpredictable demands on CPU, memory, and I/O.

The solution

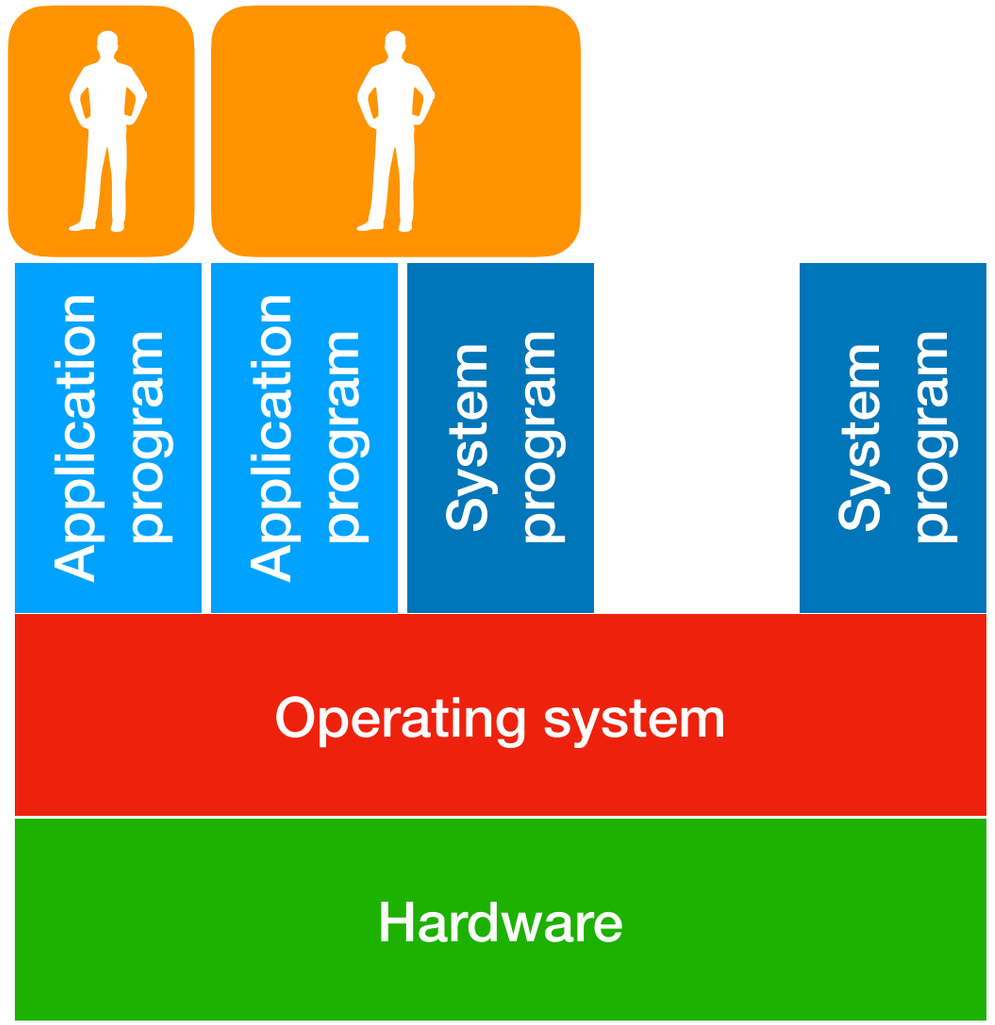
- **Programs run in isolation.** A program operates within its allocated memory space and cannot access or interfere with other running programs, ensuring stable and independent execution.

- **System resources are not directly accessible.** Programs cannot directly access system resources. Resource access is controlled and mediated, preventing unintended interference or conflicts.

- **The operating system manages all aspects of resource allocation and control!**

General purpose computing system

- **Hardware:** the physical machine that executes computation.
- **Operating system:** software that manages the computer resources.
- **Applications:** system (OS) programs and application (user) programs.
- **Users:** persons, other systems, machines…



General purpose computing system

- When a program needs a resource (CPU, memory, I/O), it requests it through the OS. The program waits for resource access before proceeding.

- The OS provides a set of system calls that allow programs to request its services.

- The OS is primarily responsible for maintaining a stable and efficient system.

- It manages resources, coordinates program access, and ensures smooth operation across all components.