Assignment - 1

1. What are Device driver?
2. Difference between general purpose system vs embedded systems
3. How can hardware understand the code we write in embedded systems (.c file to .exe file)
4. Difference between RTOS & GPOS

① What are Device driver?

Definition :-

Device drivers are software programs that act as intermediaries blw hardware device and the operating system.

Importance of Device Drivers

① Hardware compatibility :-

Device Drivers are ensure that hardware devices are compatable with the operating system. without the appropriate driver, a device may not function correctly (or) may not be recognized by the OS.

② optimized performance :-

Drivers are designed to optimize the performance of hardware devices. They can unclock advanced features improve efficiency.

③ Stability and Reliability :-

well designed drivers contribute to system stability by preventing conflices and errors blw different hardware components.

④ security :-

some drivers include security features protect against vulnerabilities.

How Device Drivers work?

① Initialization:- when the os starts it loads essential drivers into memory. These drivers include those required for system components, such as the motherboard and storage devices.

② Dynamic loading:- Additional drivers are loaded dynamically as needed when new hardware devices are detected This process is known as plug and play.

③ communication:- The os communicates with hardware devices through the drivers when a user interacts with a device, the os sends commands and data to the corresponding driver.

④ Translation:- The driver translates these commands and data into a format that the hardware understands it may involve converting high-level os commands into low level hardware instructions.

⑤ Data transfer:- Device Drivers manage the flow of data between the os and the hardware.

⑥ Error handling:- Device Drivers handle errors and exceptions that may occur during device operation.

⑦ User interface:- some Drivers provide user interfaces (or) configuration tools that allow users to customize device settings and behaviour.

Types of Device Drivers

① Kernel mode Drivers ② user mode Drivers ③ Filter drivers
④ virtual device Drivers.

Conclusion:- Device Drivers are indispensable components of modern computing systems. They enable the seamless interaction blw os and a wide range of hardware devices, ensuring, compatibility, performance optimization and sim stability understanding the role of Drivers is crucial for troubleshooting hardware issues and maintaining a smoothly functioning computer sim.

(2) Difference b/w General purpose sim and embedded sim?

(1) General purpose sim

Def:- General purpose sim like a typical personal computer is designed to perform a wide range of tasks and can run various software applications. It is flexible and adaptable with a General operating sim like windows, macos, or Linux.

(2) Embedded sim

Def:- Embedded sim is designed for a specific function or set of functions. It's often a part of a large sim of product, such as a microwave oven, automotive control sim, (or) a medical device embedded sim's have limited computing resources, run specialized software and are optimized. For reliability the efficiency.

Key differences

General purpose sim's	embedded systems.
(1) General purpose sim's serve a broad range of applications	(1) embedded sim's have specific dedicated purpose.
(2) These sim's run a variety of software applications	(2) These sim's use specialized often proprietary software.
(3) These sim's use standard hardware components	(3) These sim's often have custom hardware tailored to their specific task.
(4) These sims are flexible and can adapt to different tasks.	(4) These sim's are fixed in their function
(5) These sim's typically run Full-Featured os's.	(5) These sim's may use real time operating sim's (RTOS) or custom firmware.
(6) These sim's have ample computing resources.	(6) These sim's have limited resources to meet their specific requirements.
(7) These sim's often have a graphical user interface (GUI) for user interaction.	(7) these sims may have simple interfaces (or) non at all.

conclusion :-

General purpose sim's are versatile and capable of running various applications while embedded sim's are purpose-built for specific tasks and operates with constraints in terms of resources and functionality.

③ How can hardware understand the codes that we write in Embedded sim? (.c file to .exe file)

C file is first go through the pre-processor, then compiler compiles it to assembler and creates object file (main.o) Then linker link the main.o with required header objects and libraries and creates a executable file (program.exe)

In embedded sim's hardware understands the codes you write through a combination of components and a microcontroller / microprocessor.

Here's a simplified overview of how this works :-

① Microcontroller / microprocessor

embedded sim's typically have a microcontroller (or) microprocessor at their core. These are specialized chips designed to execute instructions They have a cpu and various peripherals.

② Machine code :-

when you write code for an embedded sim's it is usually in a high-level programming language like C (or) C++ you use a compiler to convert this high level code into machine code that the microcontroller can understand.

③ memory :-

The machine code is stored in the memory of the ES This memory can be divided into various sections, including program memory and data memory.

④ Instruction Fetch and Execution :-

The CPU of the microcontroller fetches instructions from the program memory one by one. These instructions are in the form of binary code which corresponds to specific operations.

Scanned with OKEN Scanner

(5) Peripheral interfaces :-

The microcontroller has interfaces that allow it to interact with external hardware devices.

(6) Interrupts :-

ES's often use interrupts to handle asynchronous events when an interrup occurs, the microcontroller can pause its current talk, execute a predefined interupt service routine (ISR) and then return to the main program.

(7) Clock and timing :-

The microcontroller relies on an internal clock. (or) oscillator to synchronize its operations and maintain precise timing.

(8) compiler optimization :-

compilers for ES's often provide optimization options to generate efficient code, taking into account the limited resources available in these sim's.

(9) Hardware Abstraction layers (HALS)

In some cases developers may use hardware abstraction layers (or) libraries provided by the microcontroller manufacturer to simply interaction with hardware components these libraries provide higher level functions to control hardware peripherals.

(4) what is the difference b/w RTOS & GPOS

RTOS	GPOS
Real time operating system	General purpose operating system
(1) predictable behaviour	(1) there is no predictable behaviour
(2) works under worst case assumption	(2) optimizes for Avg case.
(3) no large memory	(3) large memory
(4) priority inversion major issue	(4) priority inversion unnoticed
(5) dedicated to single network	(5) multi user environment
(6) scalable	(6) non-scalable.

(7) Time sensitive	(7) Time insensitive
(8) Flat memory model	(8) protected memory model
(9) unfair scheduling	(9) Fair scheduling
(10) scheduling based on priority	(10) adjusted dynamically
(11) Kernel is pre emptive	(11) Kernal non preemptive
(12) RTOS has got bounded latencies	(12) GPOS has unbounded latencies.
(13) Light weight and small in size	(13) Heavy and large.
(14) RTOS is designed for applications that require precise and deterministic timing often in ES's, robotics, and control sim's. It ensures that tasks are executed within specified time constraints	(14) GPOS also known as standard desktop(or) server opering sim's are designed for General purpose computing tasks. They are versatile and cater to a wide range of applications.

conclusion:-

The choice b/w RTOS and GPOS depends on the specific requirements of the sim (or) application If Determinism and real-time constraints are essential an RTOS is more suitable for general purpose computing tasks, a GPOS provides the versatility and feature set required.

Scanned with OKEN Scanner