Interview Questions

What is the use of header file?

A **header file** is generally used to define all of the functions, variables and constants contained in any function library that you might want to **use**. The **header file** stdio.h should be used if you want to **use** the two standard I/O functions printf and scanf.

One variable defined in one file and if u want to use that variable in two or more files what will you do?

Explain about storage classes in C?

Output of the fallowing c code

```
struct node
{
  int a;
  char b[];
};
void main()
{
  struct node var;
  printf(“sizeof structure=%d\n”,sizeof(var));
}
```

Ans:Error

Explain dynamic memory allocation? What is the difference between malloc() and calloc() ?

Dynamic memory allocation refers to managing system **memory** at runtime. **Dynamic memory** management in C programming language is performed via a group four functions named malloc(), calloc(), realloc(), and free().

The **malloc()** takes a single argument, while **calloc()** takess two. Second, **malloc()** does not initialize the **memory allocated**, while **calloc()** initializes the **allocated memory** to ZERO.
Both **malloc** and **calloc** are used in C language **for dynamic memory allocation** they obtain blocks **of memory dynamically**.

What is function pointer? declare function pointer to function which accepts two integers and returns float ?practical use of function pointer?

Function Pointer Syntax

In this **example**, foo is a **pointer** to a **function** taking one argument, an integer, and that returns void. It's as if you're declaring a **function** called "*foo", which takes an int and returns void; now, if *foo is a **function**, then foo must be a **pointer** to a **function**.

What is the **practical use of function pointers**? **Function pointers** can be useful when you want to create callback mechanism, and need to pass address of an **function** to another **function**. They can also be useful when you want to store an array of **functions**, to call dynamically for **example**.

<u>Write a program to find the position of first non-zero bit?</u>

<u>8. Where local,static and global variables stored in the primary memory?</u>

Questions on operating system:

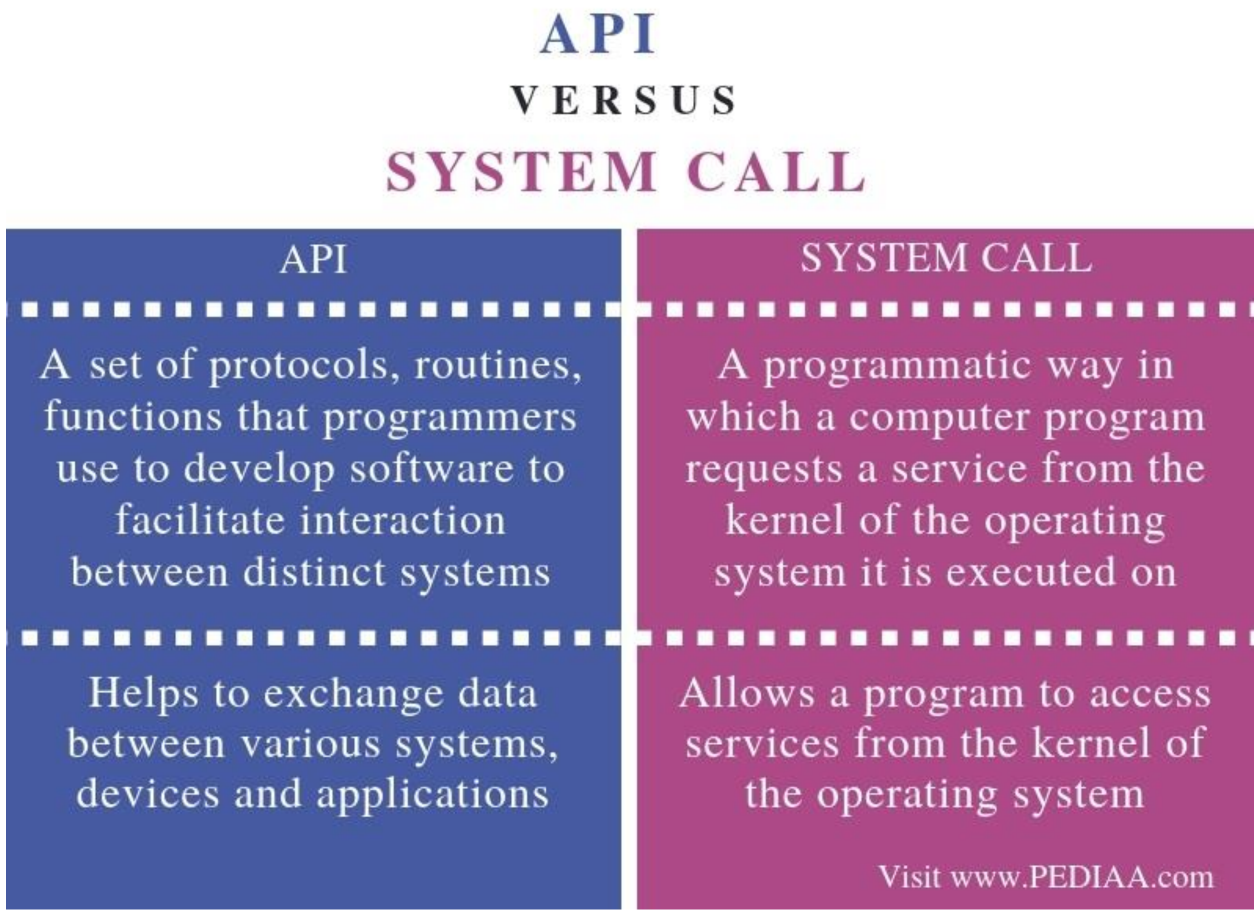
1. what is semaphore and binary semaphore?

Semaphore is simply a variable. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment. ... Counting **semaphore** can take non-negative integer values and **Binary semaphore** can take the value 0 & 1. only.

2. What is API? how system call works?

The main difference between **API** and **system call** is that **API** is a set of protocols, routines, and, functions that allow the exchange of data among various applications and devices while a **system call** is a method that allows a program to request services from the kernel.

A visual explanation of a user application invoking the open() **system call**: It should be noted that the **system call** interface (it serves as the link to **system calls** made available by the operating **system**) invokes intended **system call** in OS kernel and returns status of the **system call** and any return values.



API

VERSUS

SYSTEM CALL

API	SYSTEM CALL
A set of protocols, routines, functions that programmers use to develop software to facilitate interaction between distinct systems	A programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on
Helps to exchange data between various systems, devices and applications	Allows a program to access services from the kernel of the operating system

Visit www.PEDIAA.com

3. Explain about IPC techniques?

Inter-process communication (**IPC**) is a mechanism that allows the exchange of data between processes. ... **IPC** enables data **communication** by allowing processes to use segments, semaphores, and other methods to share memory and information. **IPC** facilitates efficient message transfer between processes.

4. What will happen when Interrupt occur?

When an **interrupt occurs**, it causes the CPU **to** stop executing the current program. ... When an **interrupt** is generated, the processor saves its execution state via a context switch, and begins executing the **interrupt** handler at the **interrupt** vector.

<u>5. How interrupt handles in OS?</u>

6. Explain open() system call and file descriptor?

For most **file systems**, a program initializes access to a **file** in a **file system** using the **open system call**. This allocates resources associated to the **file** (the **file descriptor**), and returns a **handle** that the process will use to refer to that **file**. In some cases the **open** is performed by the first access.

7. What is preemptive and Non-preemtive scheduling ?

Preemptive scheduling allows a running process to be interrupted by a high priority process, whereas in **non-preemptive scheduling**, any new process has to wait until the running process finishes its CPU cycle.

8. Explain Dead lock conditions

In an operating system, a **deadlock** occurs when a process or thread enters a waiting state because a requested system resource is held by another waiting process, which in turn is waiting for another resource held by another waiting process.

9. What is ISR?

An **ISR** (also called an interrupt handler) is a software process invoked by an interrupt request from a hardware device. ... When the **ISR** is complete, the process is resumed. A basic example of an **ISR** is a routine that handles keyboard events, such as pressing or releasing a key.

10. What is Virtual memory and Physical Memory?

Random access **memory** (**RAM**) is **physical memory** that holds the applications, documents and procedures on a computer. **Virtual memory** is a storage area that holds the files on your hard drive for retrieval when a computer runs out of **RAM**.

Questions on Embedded Systems

1. Tell me about yourself

2. What is Embedded system?

An **embedded system** is a computer **system**, made from a combination of hardware and software, that is used to perform a specific task. It may or not be programmable, depending on the application. **Examples** of **embedded** systems include washing machines, printers, automobiles, cameras, industrial machines and more

Embedded?

Embedded software is computer software, written to control machines or devices that are not typically thought of as computers, commonly known as **embedded** systems. It is typically specialized for the particular hardware that it runs on and has time and memory constraints.

3. Difference between Embedded and Non Embedded Systems?

But they differ in two key respects: The software of an **embedded system** is custom-written, to work with its specific hardware. **Non-embedded systems** are more general-purpose. **Embedded** computer software is usually loaded into **non**-volatile memory, rather than RAM.

An **embedded system** is any electronic **system** that uses a CPU chip, but that is **not** a general-purpose workstation, desktop or laptop computer. ... So presently, using a mobile phone, one cannot develop an '**embedded system**'. If it is possible by the mobile device, then it should be considered as a general purpose **system**

There might be little **difference between** the hardware and there are some **embedded systems** controlled by **PC** motherboards. ... The human interface is not to run general purpose software, but to control the machine or hardware. So the main **difference** is that purpose. If it runs general purpose software, it's a computer.

An **embedded operating system** is an **operating system** for **embedded** computer **systems**. ... Unlike a desktop **operating system**, the **embedded operating system does** not load and execute applications. This means that the **system** is only able to run a single application.

Embedded systems are combinations of hardware and software. The purpose of **embedded systems** is to control a device, a process or a larger **system**. ... As **systems** become ever more intelligent and distributed, **they** also become more complex and interdependent.

An **embedded system** is a combination of hardware and **software** that is designed to carry out a certain task or tasks, meaning it has a specific **function**. They are '**embedded**' within a larger electrical **system**.

The Role of an **RTOS** in an **Embedded System**. April 7th, 2018 by. An **embedded system** is a special computer **system** that is designed to perform dedicated functions with real-time computing constraints. It contains software, memory, and a processor that may be 8051micro-controller memory ROM or Pentium-IV processor memory ...

Advantages of embedded operating system

- Small size and faster to load.
- More specific to one task.
- Easy to manage.
- Low cost.
- Spend less resources.
- These operating system is dedicated to one device so performance is good and use less resources like memory and micro-processors.

Pre-emption is the ability of an operating system to temporarily suspend a task in order to execute a higher-priority task. If the embedded software that is being developed requires the **need** to prioritize tasks and interrupt tasks that are currently running, an **RTOS** is the go-to operating system

What are the main **features of embedded systems**? a) **Embedded systems** are application specific & single functioned; the application is known apriori, the programs are executed repeatedly. b) Efficiency is of paramount importance for **embedded systems**.

<u>4. Draw Block diagram of Embedded system and Explain Each component present in the embedded system.</u>

5. Explain about SPI and I2C Protocols and what is the difference between them? And where they used in practical?

I2C is the address base bus **protocol**, you have to send the address of the slave for the communication. In case of the **SPI**, you have to select the slave using the slave select pin for the communication. **I2C** has some extra overhead due to start and stop bits. **SPI** does not have a start and stop bits.

Serial Peripheral **Interface** (**SPI**) is an **interface** bus commonly **used** to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It **uses** separate clock and data lines, along with a select line to choose the device you wish to talk to.

I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems. It was invented by Philips and now it is **used** by almost all major IC manufacturers.

I2C requires only two wires, while **SPI** requires three or four. ... **SPI** cannot transmit off the PCB while **I2C** can, albeit at low data transmission speeds. **I2C** is cheaper to implement than the **SPI** communication protocol. **SPI** only supports one master device on the bus while **I2C** supports multiple master devices

I2C Protocol. Transmitting and receiving the information between two or more than two devices require a communication path called as a bus system. A **I2C** bus is a bidirectional two-wired serial bus which is used to transport the data between integrated circuits. The **I2C** stands for “Inter Integrated Circuit”.

I2C is generally limited to 400 KHz. But this is not really an issue for the MPU-6050/6000 accelerometer, since it runs at 400 KHz for **I2C**, and only 1 MHz for **SPI** -- not that much of a difference. In general, **SPI** is a **faster** bus - the clock frequency can be in a range of MHz

Like **I2C**, **SPI** is a **different** form of serial-communications protocol specially designed for microcontrollers to talk to each other. ... **SPI** is typically much faster than **I2C** due to the simple protocol and, while data/clock lines are shared **between** devices, each device requires a unique address wire.

6. Explain about UART Interface and what is the speed of UART?

A universal asynchronous receiver-transmitter (**UART** /ˈjuːɑːrt/) is a computer hardware device for asynchronous **serial** communication in which the data format and transmission **speeds** are configurable. The electric signaling levels and methods are handled by a driver circuit external to the **UART**.

TI claims that early models can run up to 1 Mbit/s, and later models in this series can run up to 3 Mbit/s. 128-byte buffers. This **UART** can handle a maximum standard serial port **speed** of 921.6 kbit/s if the maximum interrupt latency is 1 millisecond. This **UART** was introduced by Exar Corporation.

UART stands for Universal Asynchronous Receiver/Transmitter. It's not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC. A **UART's** main purpose is to transmit and receive serial data.

7. Which has the fast data rate among UART,USB,SPI and I2C ?

8. What is RTOS?

A real-time operating system is any operating system intended to serve real-time applications that process data as it comes in, typically without buffer delays. Processing time requirements are measured in tenths of seconds or shorter increments of time.

Difference between GPOS and **RTOS**
General purpose **operating systems** cannot perform real time tasks whereas **RTOS** is suitable for real time applications. ... Synchronization is a problem with GPOS whereas synchronization is achieved in real time kernel. Inter task communication is done using **real time OS** where GPOS does not.

9. Explain about Soft real time system and Hard Real Time system?

A **Hard Real-Time System** guarantees that critical tasks complete on **time**. ... A **Soft Real Time System** where a critical **real-time** task gets priority over other tasks and retains that priority until it completes. As in **hard real time systems** kernel delays need to be bounded.

10. Explain About LINUX Device Drivers.

The software that handles or manages a **hardware** controller is known as a **device driver**.
The **Linux** kernel **device drivers** are, essentially, a shared library of privileged, memory resident, low level **hardware** handling routines. It is **Linux's device drivers** that handle the peculiarities of the **devices** they are managing.

A **driver** provides a software interface to **hardware devices**, enabling operating systems and other computer programs to access **hardware** functions without needing to know precise details about the **hardware** being used. ... **Drivers** are **hardware** dependent and operating-system-specific.

However, **devices** and operating system cannot communicate with each other directly. **Drivers** are therefore created to help them understand each other. ... Other than that, most **devices** (like graphic card, audio card, network card) **require** a specific **driver** being installed to work properly.

Examples of utility programs are antivirus software, backup software and disk tools. A **device driver** is a computer program that controls a particular **device** that is connected to your computer.

Most **embedded** hardware requires some type of software initialization and management. The software that directly interfaces with and controls this hardware is called a **device driver**. ... **Device drivers** are the software libraries that initialize the hardware and manage access to the hardware by higher layers of software.

Generally a **driver** communicates with the **device** through the computer bus which is used to connect the **device** with the computer. ... Instead of accessing a **device** directly, an operating system loads the **device drivers** and calls the specific functions in the **driver** software in order to execute specific tasks on the **device**.

There are various **types of device drivers** for I/O **devices** such as keyboards, mice, CD/DVD drives, controllers, printers, graphics cards and ports. There are also virtual **device drivers** (VxD), which are **device driver** components that enable direct communication between a **hardware device** and an application.

11. Explain Vector Project In Detail

Questions on NETWORKING:
1. Explain about OSI reference Model and Explain Each Layer in it?

2. Why TCP/IP called so instead of calling TCP and IP individually?

The name "**TCP**/**IP**" refers to an entire **suite** of data communications protocols. The **suite** gets its name from two of the protocols that belong to it: the Transmission Control Protocol (**TCP**) and the Internet Protocol (**IP**).

3. Explain TCP/IP reference Model? Explain about Each Layer?

4. What is Protocol?

In telecommunication, a communication protocol is a system of rules that allow two or more entities of a communications system to transmit information via any kind of variation of a physical quantity

5. What is SMTP protocol and where it is Present?

SMTP Fundamentals
SMTP is an application layer **protocol**. The client who wants to send the mail opens a TCP connection to the **SMTP server** and then sends the mail across the connection. The **SMTP server** is always on listening mode.

6. Difference between TCP and UDP?

TCP (Transmission Control Protocol) is connection oriented, whereas **UDP** (User Datagram Protocol) is connection-less. This means that **TCP** tracks all data sent, requiring acknowledgment for each octet (generally). ... Because of acknowledgments, **TCP** is considered a reliable data transfer protocol.

7. Where IP protocol Present?

The Internet **Protocol** is essentially what makes the Internet different from other digital **networks** (ARPANET, for instance). **IP protocol** assigns a unique **address**, called the "**IP Address**" to each computer in a **network**, and these **IP addresses** are **used to route** packets of information from a source to a destination.

8. What is MAC Address how many bits it is?

The Format of a **MAC Address**
Traditional **MAC addresses** are 12-digit (6 bytes or 48 **bits**) hexadecimal numbers. By convention, they are usually written in one of the following three formats: MM:MM:MM:SS:SS:SS.

9. In which Layer MAC address Present?

IP **addresses** and packets are **layer** 3, **MAC addresses** are **layer** 2! Short for **Address** Resolution Protocol, a network **layer** protocol used to convert an IP **address** into a physical **address**, such as an Ethernet **address**.

HR ROUND:
Simple Questions:
1. Tell me About Yourself.
2. What you know about Redpine signals?
3. Why we hire you?

You can do the work and deliver exceptional results. **You** will fit in beautifully and be a great addition to the team. **You** possess a combination of skills and experience that make **you** stand out from the crowd. **Hiring you** will make him look smart and make his life easier.

"Honestly, **I** possess all the skills and experience that **you**'re looking for. **I**'m pretty confident that **I** am the best candidate for this job role. It's not just my background in the past projects, but also my people skills, which will be applicable in this position.

Why do you want this job?
The interviewer is looking for similar things whether asking about company or position. The hiring manager **wants** to: Learn about your career goals and how this position fits into your plan. Make sure that **you** are sincerely interested in the **job** and will be motivated to perform if hired.

What are your weaknesses answer?

When she's asked, "**What are your** greatest strengths and **weaknesses**?" Francine responds, "**My** strength is that I'm a hard worker. **My weakness** is that I get stressed when I miss a deadline because someone else dropped the ball." This **answer** is unimaginative, a no-brainer.

4. How is the Vector Training? When your course Completed?
5. What is most memorable event in your life?
6. Any questions to ask?

little endian & big endian in embedded?

Big-endian is an order in which the "**big** end" (most significant value in the sequence) is stored first (at the lowest storage address). **Little-endian** is an order in which the "**little** end" (least significant value in the sequence) is stored first.

The **big endian** format means that data is stored **big** end first. In multiple bytes, the first byte is the biggest, or represents the primary value. **In the little endian** format, data is stored **little** end first. ... Developers can use various fixes to resolve **big endian and little endian** data issues.

Is this answer still relevant and up to date? There is no meaning to say who is the better **big endian** or **little endian**, it only arranges the data in a predefined order. In the case of **little endian** you can access first bytes of data with zero offsets because LSB stored at the lower address.

Why is Little Endian better?
Consider an 8-bit system that fetches bytes sequentially from memory. If it fetches the least significant byte first, it can start doing the addition while the most significant byte is being fetched from memory. This parallelism is why performance is **better** in **little endian** on such as system

Is ARM big endian or little endian?
Current generation **ARM** processors (from ARM6 onwards) have the option of operating in either **little-endian** or **big-endian** mode. These terms refer to the way in which multi-byte quantities, such as 32-bit words, are stored in a byte-addressed memory. ... This document describes the behaviour of a **big-endian ARM**.

Is Little Endian in C?

In the above program, a character pointer **c** is pointing to an integer i. ... If machine **is little endian** then ***c** will be 1 (because last byte is stored first) and if machine is big **endian** then ***c** will be 0.

How to decide whether given processor is using little endian format or big endian format ?

The following program can find out the endianness of the processor.

```
#include<stdio.h>

main ()

{

 union Test

 {

   unsigned int i;

   unsigned char c[2];

 };

 union Test a = {300};

 if((a.c [0] == 1) &&  (a.c [1] == 44))

 {

   printf ("BIG ENDIAN\n");

 }
 else

 {

    printf ("LITTLE ENDIAN\n");

 }

}
```

Difference between RISC and CISC processor.

RISC (Reduced Instruction Set Computer) could carry out a few sets of simple instructions simultaneously. Fewer transistors are used to manufacture RISC, which makes RISC cheaper. RISC has uniform instruction set and those instructions are also fewer in number. Due to the less number of instructions as well as instructions being simple, the RISC computers are faster. RISC emphasise on software rather than hardware. RISC can execute instructions in one machine cycle.

CISC (Complex Instruction Set Computer) is capable of executing multiple operations through a single instruction. CISC have rich and complex instruction set and more number of addressing modes. CISC emphasise on hardware rather that software, making it costlier than RISC. It has a small code size, high cycles per second and it is slower compared to RISC.

Difference Between Thread & Process?

Threads are used for small tasks, whereas **processes** are used for more 'heavyweight' tasks – basically the execution of applications. Another **difference between thread and process** is that **threads** within the same **process** share the same address space, whereas different **processes** do not.

What is Multi Threading in OS?

Multithreading is the ability of a program or an **operating system** process to manage its use by more than one user at a time and to even manage **multiple** requests by the same user without having to have **multiple** copies of the **programming** running in the **computer**.

What is multithreading embedded systems?

Multithreading. **Multithreading** is mainly found in multitasking operating **systems**.**Multithreading** is a widespread programming and execution model that allows **multiple threads** to exist within the context of one process. These threads share the process's resources, but are able to execute independently.

What are the benefits of multithreading?

The **Benefits of Multithreaded** Programming. **Multithreading** allows the execution of multiple parts of a program at the same time. These parts are known as threads and are lightweight processes available within the process. So **multithreading** leads to maximum utilization of the CPU by multitasking.

what is dsr in os?

A demand signal repository (**DSR**) is a database or a data warehouse that is used for aggregating and structuring demand data. Companies use this tool to make efficient use of demand data and to anticipate change in business process automation and other planning enhancements.

what is the difference between mutex and Semaphore?

Mutex is a mutual exclusion object that synchronizes access to a resource. ... A **Mutex** is **different** than a **semaphore** as it is a locking mechanism while a **semaphore** is a signalling mechanism. A binary **semaphore** can be used as a **Mutex** but a **Mutex** can never be used as a **semaphore**.

When would you use semaphore over mutex?

Use mutex where you want to allow a piece of code (normally called critical section) to be executed by one thread at a time. **Use semaphore** to signal/notify about some event. By following few strict rules about lock/unlock a **semaphore** can be used to protect a critical section.

What is the **difference** between a **stack** and a **Queue**?

Stack is a collection of objects that works in LIFO (Last in First out) mechanism while **Queue** is FIFO (First in First out). This means that the object that is inserted first is removed last in a **stack** while an object that is inserted first is removed first in a **queue**.
Stack and Queue both are the non-primitive data structures. The **main differences between stack and queue** are that **stack** uses LIFO (last in first out) method to access and add data elements whereas **Queue** uses FIFO (First in first out) method to access and add data elements

->A **watchdog timer** (WDT) is a hardware **timer** that automatically generates a system reset if the main program neglects to periodically service it. It is often used to automatically reset an embedded device that hangs because of a software or hardware fault.

->> **Static linking** is performed by programs called linkers as the last step in compiling a program. Linkers are also called **link** editors. **Dynamic linking** is performed at run time by the operating system. **Statically** linked files are significantly larger in size because external programs are built into the executable files.