LTI Recruitment Process

1. Interview Process

LTI examines a candidate thoroughly throughout the hiring process. The candidates at LTI need to undergo the following rounds and pass in all of them to be considered for the job opening:

- Online Assessment
- Technical Interview Rounds
- HR Rounds

All of the above rounds are eliminative and you must successfully clear all of them to be considered for the role.

2. Interview Rounds

Round 1-Online Assessment: The first stage in the hiring process is to take an online assessment test. Those who pass this round will be invited to the technical interview rounds. The exam is divided into seven sections.

The following is a detailed online test pattern for the LTI recruitment process:

- **Quantitative Aptitude MCQ (10 questions/10 minutes):** This part has 10 medium-level questions about profit and loss, ages, time and work, speed, distance, and time, among other topics. If you are good at Mathematics, this section will be a piece of cake for you. You can brush up on fundamental mathematics for this section.
- **MCQs on logical reasoning (10 questions/10 minutes):** This part has ten medium-level questions about blood relations, directions, artificial language, visual reasoning, syllogism, and other topics. You just need to be alert and calm while solving these questions. Also, time management in these types of questions plays a key role. Do not spend too much time on a single question. If you are stuck on a particular question, consider moving to the next question. You might revisit the question later, if time permits.
- **Verbal Reasoning MCQ (10 questions/10 minutes):** This section generally contains 10 questions of a moderate difficulty level. The questions are centred on prepositions, basic grammar such as error detection, and choosing one word for a sentence, among other things.
- **Technical MCQ (20 questions, 20 minutes):** This section generally contains 20 questions that are similarly of a low difficulty level and are designed to assess your basic technical knowledge. You can expect questions from CS fundamental subjects like Object Oriented Programming (OOPs), DataBase Management Systems (DBMS), Operating Systems (OS), Computer Networks (CN) and so on. Prepare these subjects well and brush up on the concepts before you appear for the assessment.
- **Paragraph Writing (1 topic/10 minutes):** In this section, you will be given a topic on which you must write a 200-word paragraph. You don't need to prepare anything for this section, but you should have some basic English writing skills to assist you to pass it.
- **Psychometric Questions (86 MCQs/ 20 minutes):** In this section, you will be generally asked situational based questions. This section assesses the moral values and the character of the

candidate to make sure that it aligns with that of LTI. You do not need to prepare for this section. Just be true to yourself.

- **Coding Question (1 question/ 45 minutes):** This section generally consists of 1 coding question to be solved in 45 minutes. The difficulty level of the question is generally medium. You should have a thorough practice of Data Structures and Algorithms for this section. You can also use our programming section to prepare for this round. On each topic, we've compiled a detailed list of frequently asked coding questions. This will assist you in answering all of the crucial coding questions before taking a coding assessment. This link will take you to our programming section.

Round 2 - Technical Interview Rounds: Candidates who clear the online assessment are called for technical interview rounds. Your technical skills are evaluated during a technical interview, which is usually related to the technical knowledge required for the position and the company you wish to work for. The questions will put your problem-solving and numerical reasoning skills to test. As a result, in the Technical Round, the interviewer will assess you based on your:

- Technical Expertise and problem-solving skills.
- IQ Quotient
- Projects and your past work experiences

The technical face-to-face interview is the most crucial step in the process. You should be familiar with computer science concepts such as Object-Oriented Programming (OOPs), DataBase Management Systems (DBMS), Operating Systems (OS), Computer Networks (CN), and be able to explain them to the interviewer.

It is necessary to be familiar with a programming language. Make sure you know at least one programming language. You do not need to be fluent in all programming languages, but at least one, such as C++, Java, or Python, should be acquainted with you. Your problem-solving talents will also be evaluated by the interviewer. You'll be asked about your previous projects and work experiences, including what you did, what technology you utilized, what you developed, and how successful you were. There may also be questions concerning the CV and the firm.

Candidates at LTI usually need to undergo 1 round of technical interviews. However, the number of technical round interviews may vary depending on your performance in the first round of interviews.

Round 3 - HR Interview Round: The HR round will be held for students who have passed the technical interview. **HR interviews** are used to identify whether or not an applicant is qualified for the position and how well they will fit into the company's culture. Expect questions on work experience, educational qualifications, interests, and family background, in addition to the standard HR questions about strengths and flaws, reasons for applying to the company, why you should be recruited, and so on. They can also inquire about LTI's history, including when it was started, its objectives, principles, and goals, as well as its organizational structure.

Most candidates assume that the HR interview is simple, but keep in mind that even if you have cleared all other hurdles, a poor HR interview can endanger your chances of getting the job.

Maintaining a nice and confident demeanor is the goal. Remember to smile throughout interviews because they can be long and tedious.

The following are the most frequently asked questions:

- Tell me about yourself. You can start with your family history, then go on to your educational degrees, and finally your career development. When speaking about yourself, try to incorporate specific details that will persuade the interviewer that you are qualified for the job.
- What are some of your strengths and weaknesses? Be open and honest about your strengths and weaknesses. You may have several strengths but focus on the ones that will help you succeed in this role. Make your weakness sound like your strength to a significant extent. As soon as you recognise your weakness, state how you want to overcome it.
- What is your motivation for wanting to work at LTI? Here, mention the things that motivate you to work at LTI. This could include the tech stack used at LTI, the perks of working at LTI and so on.
- Please furnish me with any LTI-related information you have. Here, the interview wants to assess the level of information you have for the company you are applying for. This reflects your desire to work at the company. Make sure to know about LTI before you appear for their interview.
- What value do you bring to LTI, and how do you envision yourself contributing to the world while you're here? For these types of questions, make sure to go through the mission and vision of the company. This will help you to frame your answer aligning with the principles of the company.
- Is it possible for you to relocate to another part of the country? Here, you should clearly express whether you are comfortable or not with relocation..
- What was it about LTI that intrigued you in the first place? Here, you can talk about what motivates you to work at LTI and why is it your first choice.
- What are your expectations regarding your salary at LTI? Here, do not answer directly that you expect X salary. Instead, you can give answers like “Since LTI is an established company, I believe that the salary would range from X.”

LTI Technical Interview Questions: Freshers and Experienced

1. What are the differences between C and C++ programming languages?

- **C programming language** - C is a machine-independent structural or procedural-oriented computer language that is widely utilized in a variety of applications. C is a fundamental programming language that can be used to create everything from operating systems (such as Windows) to complicated applications such as the Oracle database, Git, Python interpreter, and many others. Because it serves as the foundation for other programming languages, the C programming language might be referred to as a god's programming language. We can simply learn other programming languages if we already know C. Dennis Ritchie, a legendary computer scientist, created the C programming language at Bell Laboratories. It has a few extra characteristics that set it apart from other programming languages.

- **C++ programming language** - Bjarne Stroustrup created C++ in 1980 at Bell Labs as a special-purpose computer language. C++ is a programming language that is very similar to C and is so compatible with C that it can run 99 per cent of C programs without modifying any source code. However, because C++ is an object-oriented programming language, it is a safer and more well-structured programming language than C.

2. What are some of the advantages and disadvantages of Object-Oriented Programming languages?

The following are the advantages of Object Oriented Programming languages:

- Instead of needing to start writing code from scratch, we can build programs from standard functioning modules that communicate with one another, which saves time and increases productivity. The OOP language allows us to divide the program down into bite-sized problems that can be solved quickly (one object at a time).
- Object-Oriented Programming languages have proven to increase programmer efficiency, improve software quality, and reduce maintenance costs.
- It is feasible for several instances of a class to coexist without interfering with each other. This allows for parallel development.
- It is relatively simple to divide a project's work into objects.
- We can minimise duplicate code and increase the use of existing classes by using inheritance.
- The data concealing principle helps programmers in developing secure programs.
- More information about the model in an implementable form can be captured by using the data-centred design technique.

The following are the disadvantages of Object-Oriented Programming languages:

- The length of programs written in the OOP language is significantly greater than that of programs written in the procedural technique. As the program grows in size, it takes longer to execute, resulting in slower program execution. That can make it be an inefficient choice when there are technical limitations involved due to the size that it can end up being. Because of the duplication involved, the first-time coding can be more extensive than other options as well.
- Because using OOP is a little challenging, programmers must have excellent design and programming skills, as well as adequate planning.
- It takes some time to become used to OOPs. For some people, the thinking process involved with object-oriented programming is not natural.

3. What is the difference between method overloading and method overriding in the context of the Java programming language?

- **Method overloading** - Method Overloading is a polymorphism that occurs at compile time. Method overloading is when a class has multiple methods with the same name but distinct signatures. The return type of a method can be the same or different in method overloading, but we must alter the argument since we cannot create method overloading in Java by just changing the method's return type.

Method Overloading	Method Overriding
Method overloading is a polymorphism that occurs at compile time.	Method overriding is a polymorphism that occurs at runtime.
It contributes to the program's readability.	It's used to provide the method's specific implementation, which is already provided by its parent or superclass.
Method overloading is done within a single class.	Method overriding is done in two different classes which have the relationship between them through inheritance.
Inheritance may or may not be required for method overloading.	Inheritance is always required for method overriding.
Methods must have the same name but distinct signatures when overloading.	Methods that are overridden must have the same name and signature.

4. What do you understand about exceptions in the context of the Java programming language? What are the different types of exceptions in Java?

An exception is an unwelcome or unexpected occurrence that occurs during the execution of a program, i.e. at run time, and disturbs the program's usual flow of instructions. The application can detect and handle exceptions. When a method throws an exception, it creates an object. The exception object is the name given to this object. It contains details about the exception, such as the name and description of the error, as well as the program's state at the time the error occurred.

An exception can happen for a variety of reasons. Here are a few examples:

- User input that is not valid.
- Failure of a device.
- A network connection is lost.
- Physical restraints (eg: out of disk memory).
- Code flaws.
- Attempting to open a file that is currently unavailable.

Java defines several different types of exceptions for its various class libraries. Users can also define their exceptions in Java. The following are the different types of exceptions in Java:

- **Built-in Exceptions:** Exceptions that are present in Java libraries are known as built-in exceptions. These exceptions are appropriate for explaining specific mistake scenarios.
  - **Checked Exceptions**: Checked exceptions are also known as compile-time exceptions since the compiler checks them at compile time.
  - **Unchecked Exceptions**: Unchecked exceptions are the polar opposite of checked exceptions. These exceptions will not be checked by the compiler at compilation time. To put it another way, if a programme throws an unchecked exception, even if we don't handle or declare it, the programme won't throw a compilation error.

- **User-Defined Exceptions**: In some cases, Java's built-in exceptions are unable to adequately explain a scenario. In such instances, users can design their exceptions, known as 'user-defined Exceptions'.

5. What do you understand about Abstraction in the context of Object-Oriented Programming languages? What are its advantages?

Abstraction Is obfuscating the internal implementation and focusing solely on the services. It is accomplished by utilising abstract classes and interfaces and then putting them into action. Only the features of an object that distinguish it from all other objects are required. Only the most crucial details are highlighted, while the remainder is hidden from the user or reader.

Let us consider an example of abstraction in the real world: By emphasizing the set of services offered by the bank via the ATM GUI screen, the bank is highlighting the set of services offered by the bank without highlighting internal implementation.

Abstraction can be divided into three categories as follows:

- **Procedural Abstraction:** As the name implies, procedural abstraction entails a series of procedures in the form of functions that are followed one after the other to achieve abstraction through classes.
- **Data Abstraction:** As the phrase implies, abstraction is the process of extracting information from a set of data that describes an entity.
- **Control Abstraction:** Control abstraction is achieved by creating the program in a way that encloses object details.

The following are the advantages of abstraction:

- The main benefit of using abstraction in programming is that it allows you to group several related classes as siblings
- Because there are no highlights to internal implementation, users or communities can accomplish security.
- The enhancement will become easier because any modifications to the internal system may be made without affecting end users.
- It gives the end-user additional flexibility in how they can use the system.
- It expands the application's functionality.

6. What do you understand about virtual functions in the context of the C++ programming language?

A virtual function is a member function defined in a base class that is redefined (overridden) by a derived class. When you use a pointer or a reference to the base class to refer to a derived class object, you can call a virtual function for that object and have it run the derived class's version of the function. Regardless of the type of reference (or pointer) used for the function call, virtual functions ensure that the right function is called for an object. They're mostly used to achieve

polymorphism at runtime. The virtual keyword is used to declare functions in base classes. The call to a virtual function is resolved at runtime.

Let us understand it better with the help of the following example:

```
#include<bits/stdc++.h>
using namespace std;

class Parent {
public:
   virtual void fun() // virtual function
   {
       cout << "Inside the fun() method of Parent class\n";
   }

   void foo()
   {
       cout << "Inside the foo() method of Parent class\n";
   }
};

class Child : public Parent {
public:
   void fun()
   {
       cout << "Inside the fun() method of Child class\n";
   }

   void foo()
   {
       cout << "Inside the foo() method of Child class\n";
   }
};

int main()
{
   Parent *parent_object_pointer;
   Child child_object_pointer;
   parent_object_pointer = &child_object_pointer;

   // Virtual function, binded at runtime
   parent_object_pointer->fun();

   // Non-virtual function, binded at compile time
   parent_object_pointer->foo();

   return 0;
}
```

Output:

```
Inside the fun() method of Child class
Inside the foo() method of Parent class
```

Explanation:

In the above code, the class Child inherits from the class Parent. Two methods fun() and foo() are defined in the Parent class and are overridden in the Child class. However, the fun() method is declared as a virtual function by using the 'virtual' keyword. Inside the main() method, we create an object of the Parent class and initialise it with an object of the Child class. So, the call to the fun() method is resolved at run time and not at compile time. Because of this, the statement "Inside the fun() method of Child class" gets printed. Similarly, the call to the foo() method gets resolved at compile time and not at run time. Because of this, the statement "Inside the foo() method of Parent class" gets printed.

7. What is the difference between functions and procedures in the context of computer programming?

- **Function:** One of the most fundamental concepts in computer programming is function. It's used to compute anything based on a set of inputs. User-defined or pre-defined functions are also possible. A block of code in the function program performs certain duties or functions.
- **Procedure:** A procedure is a set of instructions or commands used in computer programming. It is referred to as a procedure, subroutine, function, or subprogram depending on the programming language. For example, in SQL both function and procedures are distinct concepts.

The following table lists the differences between functions and procedures:

Function	Procedure
A function is a type of equation that is used to calculate anything from a set of inputs. As a result, it got its name from Mathematics.	A procedure is a collection of commands that are executed in a specific order.
In SQL, a procedure can invoke a function.	A function, on the other hand, cannot invoke a procedure in SQL.
We can't use DML (Data manipulation language) commands like Insert, Delete, or Update inside a SQL function.	DML instructions can be used inside the procedure in SQL.
SQL queries can call functions.	Procedures, however, cannot be accessed via a SQL query.
When functions are called, they are compiled each time.	Procedures, on the other hand, are only compiled once and can be called as many times as needed without having to be compiled each time.
A function's return statement returns the control and result value to the calling code.	The procedure's return statement returns control to the calling code, it cannot return the result value.
Explicit transaction handles are not supported by the function.	Explicit transaction handles are supported by the procedure.

Function	Procedure
The SELECT statement can be used to work on functions.	It is not possible to use it in a SELECT statement.

8. What is the difference between mutable and immutable objects in the context of the Java programming language?

- **Mutable objects** - Objects that can have their value altered after initialization are known as mutable objects. After the object is formed, we can update its values, such as fields and states. Java.util.Date, StringBuilder, and StringBuffer are just a few examples. When we update the value of an existing mutable object, no new object is generated; instead, the value of the existing object is changed. The classes of these objects contain methods for making modifications to them.
- **Immutable objects** - Immutable objects are those whose values cannot be modified once they have been created. We are unable to make any changes to the object once it has been created. Primitive objects such as int, long, float, and double, as well as all legacy classes, Wrapper class, and String class, are examples. In a nutshell, immutable refers to something that cannot be changed or modified. The object values and state of immutable objects cannot be modified once they have been created. For immutable objects, only Getters (get() method) and not Setters (set() method) are available.

The following table lists the differences between mutable objects and immutable objects:

Mutable Objects	Immutable Objects
Without creating a new object, mutable objects can be altered to any value or state.	Immutable objects, on the other hand, cannot have their value or state modified after they have been formed. Whenever we modify the state of an immutable object, a new object is produced.
Mutable objects have a method for changing the object's content.	Immutable objects, on the other hand, do not have any methods for changing their values.
Setters and getters are both supported by mutable objects.	Immutable objects, on the other hand, only support setters and not getters.
Mutable objects may or may not be thread-safe.	Immutable objects are thread-safe by default.
StringBuffer, Java.util.Date, StringBuilder, and other mutable classes are examples.	Legacy classes, wrapper classes, String classes, and other immutable objects are examples.

9. What do you understand about checkpoints in the context of Database Management Systems (DBMS)? What are the advantages of using checkpoints?

The checkpoint specifies a time when the DBMS was in a consistent state and all transactions had been committed. These checkpoints are tracked during transaction execution. Transaction

log files will be created after the execution. The log file is destroyed when it reaches the savepoint/checkpoint by recording its update to the database. Then a new log is produced with the transaction's upcoming execution actions, which is updated until the next checkpoint, and the process continues.

When transaction logs are created in a real-time setting, they consume a significant amount of storage space. Keeping track of each update and maintaining it may also take up more physical space on the system. As the transaction log file grows in size, it may eventually become unmanageable. Checkpoints can be used to address this. A Checkpoint is a way for deleting all prior transaction logs and saving them in a permanent storage location.

The following are the advantages of checkpoints:

- It makes the data recovery procedure go faster.
- The majority of DBMS packages perform self-checkpoints.
- To avoid unnecessary redo operations, checkpoint records in the log file are employed.
- It has very low overhead and can be done frequently because dirty pages are flushed away continuously in the background.

10. What do you know about RDBMS? What are the advantages and disadvantages of using RDBMS?

RDBMS is an acronym for **Relational DataBase Management Systems**. It's an application that lets us build, remove, and update relational databases. A relational database is a database system that stores and retrieves data in the form of rows and columns in a tabular format. It is a minor subset of DBMS that was created in the 1970s by E.F Codd. The major DBMS, such as SQL, My-SQL, and ORACLE, are all based on relational DBMS concepts.

The following features are simulated by Relational Database Management Systems to maintain data integrity:

- **Entity Integrity:** No two records of a database table can be completely duplicated.
- **Referential Integrity**: Only those rows of those tables that are not used by other tables can be erased. Otherwise, data discrepancy may result.
- **User-defined Integrity:** Rules based on confidentiality and access that are established by the users.
- **Domain integrity:** The database tables' columns are contained inside certain specified limits based on default values, data types, or ranges.

The following are the advantages of an RDMS:

- **Simple to manage:** Each table can be modified independently without impacting the others. It offers logical database independence i.e. data can be viewed in different ways by the different users.

- **Secured**: It is more secure since it has numerous levels of security. Access to shared data might be restricted.
- **Flexible:** Data may be updated at a single location without having to make changes to many files. It limits redundancy and replication of the data. Databases can be simply expanded to accommodate additional records, increasing scalability. It also makes it easier to use SQL queries.
- **Users:** RDBMS can store many users in a client-side architecture.
- Allows enormous amounts of data to be stored and retrieved with ease.
- **Simple Data Handling:** The relational architecture allows for faster data retrieval. Due to keys, indexes, and normalisation principles, data redundancy or duplication is avoided. Because RDBMS is built on ACID principles for data transactions, data consistency is ensured (Atomicity Consistency Isolation Durability).
- **Tolerance for Failures**: Database replication allows for simultaneous access and aids in system recovery in the event of disasters such as power outages or unexpected shutdowns.

The following are the disadvantages of an RDMS:

- **High Cost and Extensive Hardware and Software Support:** To make these systems work, huge budgets and setups are necessary.
- **Scalability:** As more data is added, more servers, as well as more power and memory, are necessary.
- **Complexity:** Large amounts of data complicate understanding of relationships and may reduce performance.
- **Structured Limits:** A relational database system's fields or columns are surrounded by various limits, which can lead to data loss. It is difficult to recover the lost data.

11. What do you understand by query optimisation in the context of SQL?

A single query can be run using various methods or rewritten in various formats and structures. As a result, the topic of query optimization arises: which of these forms or paths is the most efficient? The query optimizer considers all alternative query plans to discover the most efficient way to execute a query.

The purpose of query optimization is to reduce the number of system resources necessary to complete a query, allowing the user to receive the proper result set more quickly.

- It gives the user faster results, making the application appear to be speedier.
- Second, because each request takes less time than unoptimized queries, the system can handle more queries in the same amount of time.
- Finally, query optimization decreases the amount of wear on the server's hardware (for example, disk drives) and allows it to run more efficiently (e.g. lower power consumption, less memory usage).

A query can be optimized in one of two ways:

- **Analyze and alter relational phrases that are equivalent:** Reduce the number of tuples and columns in the intermediate and final query processes (discussed here).
- **For each operation, multiple algorithms can be used:** These underlying techniques influence the frequency of disk and block accesses by determining how tuples are accessed from the data structures they are stored in, indexing, hashing, and data retrieval.

12. What do you understand about ACID properties in the context of Database Management Systems (DBMS)?

- **Atomicity:** Atomicity refers to the fact that data remains atomic. It means that if any operation on the data is conducted, it should either be performed or executed completely, or it should not be performed at all. It also implies that the operation should not be interrupted or just half completed. When performing operations on a transaction, the operation should be completed rather than partially.
- **Consistency:** The term "consistency" denotes that the value should be retained at all times. The integrity of the data should be maintained in DBMS, which means that whenever a modification is made to the database, it should always be preserved. The integrity of the data is critical in transactions so that the database remains consistent before and after the transaction. The information should always be accurate.
- **Isolation:** The term "isolation" refers to the act of being separated from others. Isolation in DBMS refers to the property of a database where no data from one database should impact the other and where many transactions can take place at the same time. In other words, when the operation on the first database is finished, the process on the second database should begin. It indicates that if two actions are conducted on two different databases, the value of one database may not be affected by the value of the other. When two or more transactions occur at the same time in the case of transactions, consistency should be maintained. Any modifications made in one transaction will not be visible to other transactions until the change is committed to the memory.
- **Durability:** The term "durability" refers to something's ability to last. The term durability in DBMS refers to the fact that if an operation is completed successfully, the data remains permanent in the database. The database's durability should be such that even if the system fails or crashes, the database will survive. However, if the database is lost, the recovery manager is responsible for guaranteeing the database's long-term viability. Every time we make a change, we must use the COMMIT command to commit the values.

13. What do you understand about Data Independence in the context of DataBase Management Systems (DBMS)?

In Database Management Systems (DBMS), data independence implies that the application is independent of the storage structure and data access mechanism. It allows you to change the schema definition at one level without having to change the schema definition at the next level up.

Data independence can be divided into two categories:

- **Physical Data Independence:** The data contained in the database is referred to as physical data. It's in a binary format. Physical changes should not have an impact on the logical level. For example, if we wish to edit data within a table, we shouldn't change the table's format.
- **Logical Data Independence (LDI):** The ability to update the conceptual schema without having to change the external schema is referred to as logical data independence. To distinguish the external level from the conceptual view, logical data independence is used. Any modifications to the conceptual representation of the data will not affect the user's view of the data.

14. What do you understand about deadlocks in the context of Operating Systems? What are the necessary conditions for deadlock to occur in a system?

When a process enters a waiting state because another waiting process is holding the requested resource, it is referred to as a deadlock. A typical difficulty in multi-processing is a deadlock, which occurs when numerous processes share a mutually exclusive resource known as a soft lock or software.

The following are the necessary conditions for a deadlock to occur in a system:

- **No preemption** - A resource can only be released freely by the process that is holding it once it has completed its task.
- **Mutual Exclusion** - The resource should not be shareable. If the resource is being used by a process, it cannot be used by another process until the current process releases it.
- **Hold and wait** - Suppose that a process holds onto some of the resources required for its execution. However, some of the resources that it requires are being held by other resources. Deadlock can happen only when this process does not give away its acquired resources and keeps on waiting for the other resources.
- **Circular Wait** - Two or more processes waiting for each other to complete their task and release the resources should form a cyclic dependency graph. It is only then that no process would be able to complete its execution and a deadlock would be obtained.

15. What is the difference between deadlock and starvation in the context of Operating Systems?

Deadlock - When each process possesses a resource and waits for another process to retain a resource, a deadlock ensues. Mutual Exclusion, Hold and Wait, No Preemption, and Circular Wait are all necessary circumstances for the deadlock to occur. No process that is holding one resource while waiting for another is performed in this scenario.

Starvation - When high priority processes continue to run while low priority processes are stalled for an indeterminate time, this is known as starvation. A continual stream of higher-priority processes in a densely loaded computer system can prevent a low-priority operation from ever receiving the CPU. When resources are scarce, high-priority processes consume them continuously. Ageing can help to tackle the problem of starvation. The importance of long waiting processes is increasingly raised as processes age.

Deadlock	Starvation
In the case of deadlock, none of the processes is executed since they are waiting for each other to finish.	In case of starvation, high-priority processes continue to run, whereas low-priority activities are halted.
In this case, processes are preventing resources from being used.	In this case, high-priority processes consume resources constantly.
The necessary conditions for deadlock are: No Preemption Hold and Wait Circular Wait Mutual Exclusion	The necessary condition for starvation includes that the process must be having low priority and there must be other high priority resources that are continuously occupying the resources.
It can be avoided by avoiding the conditions that lead to deadlock.	Starvation can be avoided with the concept of ageing.

16. What do you understand about kernel in the context of Operating Systems?

The kernel is a component of an operating system that manages computer and hardware functions. It primarily oversees memory and CPU functions. It's a crucial part of any operating system. Kernel serves as a link between applications and hardware-based data processing via inter-process communication and system calls.

When an operating system is loaded, the kernel is loaded initially and remains in memory until the operating system is shut down. It's in charge of things like disk management, task management, and memory management.
It determines which processes should be assigned to the CPU and which should be kept in the main memory for execution. It serves as a link between user software and hardware. The kernel's primary goal is to manage communication between software, such as user-level applications, and hardware, such as the CPU and disc memory.

The following are the different types of the kernel in Operating Systems:

- **Monolithic Kernel** - A monolithic kernel is one in which all operating system services run in kernel space. There are interdependencies between the components of the system. It has a large number of lines of code that is difficult to understand.
- **MicroKernel** - This is a sort of kernel that takes a minimalist approach. It has thread scheduling and virtual memory. With fewer services in kernel space, it is more stable.
- **Hybrid Kernel** - This is a hybrid kernel that combines both monolithic and microkernel functionality. It combines the monolithic kernel's performance and design with the modularity and stability of a microkernel.

- **Exo Kernel** - Exo Kernel is a form of the kernel that adheres to the end-to-end philosophy. It uses the fewest possible hardware abstractions. It assigns physical resources to different applications.
- **Nano Kernel** - This is a kernel that provides hardware abstraction but does not provide system services. Because the MicroKernel lacks system services, the Microkernel and Nano Kernel have become interchangeable.

LTI Interview Preparation

1. Interview Preparation Tips

Readers of this post should strongly consider the following tactics for preparing for an interview at LTI:

- Put yourself to the test to see how quickly you can solve a set of coding problems. This will help you improve your problem-solving skills and reasoning strategies.
- Share your experiences and be ready to demonstrate leadership, teamwork, professional and academic success, communication skills, and problem-solving abilities.
- Maintaining a positive and welcoming attitude is always beneficial. Introduce yourself confidently to start the conversation on a favourable note.
- Learn more about the company. Understand the company's objective and philosophy, as well as its services and product lines. Employers want you to understand the company, what it does, and how it connects to your career objectives.
- The more you prepare for the interview, the more likely you are to succeed. Acquire a thorough understanding of interview stages, rounds, and questions, among other things. Prepare answers to common HR and management interview questions in advance. Pay close attention to your technical subjects and the final project.
- Examine your resume to make sure you've included all relevant information about yourself, and that the information you've provided is correct to the best of your knowledge. Prepare to answer any questions that may be asked based on your resume.
- Make sure you're up to date on the most recent technologies. You should have a basic awareness of contemporary technological trends including artificial intelligence (AI), big data, and other related issues. Understand the fundamentals of computer science and programming languages.

Frequently Asked Questions

1. Is LTI technical interview easy?

Technical Interviews at LTI are relatively easy as compared to other companies. Generally, candidates are required to undergo only 1 round of technical interviews which is generally not the case with other companies. That being said, one should not take an interview at LTI lightly. Only when you have the right preparation and guidance you will be able to crack the interviews.

2. How many rounds in LTI are for the experienced?

There are generally three rounds of assessments for the job role of a software engineer at LTI. These assessments include the following:

1. Online Assessment.
2. Technical Interview Round.
3. HR Interview Round.

3. Why do you want to join LTI?

This is a behavioural-based question. Here the interviewer wants to assess your interests and your motive to join the company. You should clearly state what motivates you to join this particular company. This could include the work culture of the company, the technologies on which the company works and so on.

A sample answer to this could be: *"LTI is one of the leading private organisations that is contributing to India's development. I have heard from my seniors that LTI uses a vast variety of advanced technologies in their work. I strongly believe that joining LTI as a software engineer will help me to upskill myself and I would experience a steep learning curve if I got this opportunity."*

4. How do you introduce yourself in an interview?

You will need to introduce yourself in every interview. This lets the interviewer know a brief about you. Interviewers also assess you while you speak. So, you must be confident while you introduce yourself. The introduction should include your name, the place where you are residing currently, the college you are in and the degree you are pursuing, what have been your interests throughout your college years and lastly you can also include what are your hobbies. Make sure that you do not elongate your introduction unnecessarily. The introduction should be short, crisp and to the point.