技术实现类6-10

面试问答题（中英文）

技术实现类（6-10）

6、什么是EIP-1559?

What is EIP-1559?

答：

Ethereum Improvement Proposal (EIP) 1559是以太坊的一个升级，旨在改变以太坊计算和处理网络交易费用（称为“gas费用”）的方式。该升级通过使用基于区块的基础费用和发送方指定的最大费用，而不是对gas价格进行竞价，来更加平衡地激励矿工在高或低网络拥塞期间进行挖矿，从而使以太坊交易更加高效。EIP-1559是一个提案，它改变了gas费用的结构和矿工的奖励方式。该提案于2021年8月5日作为以太坊伦敦硬分叉的一部分实施。gas费 = 基础费（Basefee）+ 矿工费（Tip）。基础费会根据区块的Gas利用率动态调整，如每个区块的Gas费利用率低于50%，就降低手续费，高于50%，就提高手续费。

Ethereum Improvement Proposal (EIP) 1559 is an upgrade to Ether that aims to change the way Ether calculates and processes network transaction fees (called "gas fees"). The upgrade makes Ether transactions more efficient by using a block-based base fee and a sender-specified maximum fee, rather than bidding on the price of the gas, to provide a more balanced incentive for miners to mine during periods of high or low network congestion. EIP-1559 is a proposal to change the way the gas fee is structured and how miners are rewarded. The proposal was implemented on August 5, 2021 as part of the London Hard Fork of Ether. gas fee = base fee (Basefee) + miner fee (Tip). The base fee will be dynamically adjusted based on the Gas utilization of the blocks, such that if the Gas fee utilization of each block is below 50%, the fee will be reduced, and if it is above 50%, the fee will be increased.

7、ERC20 中的 transfer 和 transferFrom 之间有什么区别?

What is the difference between transfer and transferFrom in ERC20?

答：

transfer是从当前合约转账给目标账户，transferFrom是可设置发送账户和目标账户

transfer(address recipient, uint256 amount)

transferFrom(address sender, address recipient, uint256 amount)

Transfer is to transfer money from the current contract to the target account, transferFrom is to set the sender account and the target account.

transfer(address recipient, uint256 amount)

transferFrom(address sender, address recipient, uint256 amount)

8、如何向没有payable 函数、receive 或回退的合约发送以太坊?

How to send eth to a contract that has no payable function, receive or fallback?

答:

①通过其他合约自毁将自毁的eth发送目标合约

②将挖矿获取的区块奖励费用的接收者设置为目标合约地址

③将Beacon信标链上质押后的提现地址设置为目标合约地址

这里补充一个：在solidity0.4之前的合约，也可以直接接收ETH。

①Send the self-destructed eth to the target contract by self-destructing another contract.

②Set the recipient of the block reward fee from mining to the target contract address.

③Set the address of the Beacon chain that will be used for withdrawals after pledging as the target contract's address.

Here is one more thing: the contracts before solidity0.4 can also receive ETH directly.

9、代理合约中的存储冲突是什么?

What are storage conflicts in proxy contracts?

答:

存储冲突是指多个合约尝试访问同一存储位置时发生的问题。当多个合约同时尝试更新同一存储位置时，可能会发生存储冲突，导致数据不一致或合约无法正常工作。为了避免这种情况，使用代理合约来管理存储位置，并确保只有一个合约可以访问每个存储位置。代理合约可以充当存储位置的所有者，并使用访问控制机制来限制对存储位置的访问。这可以帮助确保数据的一致性，并提高智能合约的安全性和可靠性。

A storage conflict is a problem that occurs when multiple contracts try to access the same storage location. When multiple contracts try to update the same storage location at the same time, a storage conflict may occur, resulting in inconsistent data or a contract that does not work properly. To avoid this, use proxy contracts to manage storage locations and ensure that only one contract can access each storage location. A proxy contract can act as the owner of a storage location and use access control mechanisms to restrict access to the storage location. This can help ensure data consistency and improve the security and reliability of smart contracts.

10、abi.encode 和 abi.encodePacked 之间有什么区别?

What is the difference between abi.encode and abi.encodePacked?

答:

abi.encode会在每个参数之间添加填充，以确保每个参数占用32个字节。这可以确保编码后的字节数组具有固定的长度，并且可以正确地解码回原始参数。但是，由于填充的存在，编码后的字节数组可能会比实际需要的更大。

abi.encodePacked不会添加填充，而是将所有参数拼接在一起。这可以确保编码后的字节数组尽可能小，但是可能会导致解码时出现问题，因为无法确定每个参数的确切位置。

Abi.encode adds padding between each parameter to ensure that each parameter takes up 32 bytes. This ensures that the encoded byte array has a fixed length and can be correctly decoded back to the original parameters. However, due to padding, the encoded byte array may be larger than it actually needs to be.

Instead of adding padding, abi.encodePacked stitches all the parameters together. This ensures that the encoded byte array is as small as possible, but may lead to problems when decoding because it is not possible to determine the exact position of each parameter.

Instead of adding padding, abi.encodePacked stitches all the parameters together. This ensures that the encoded byte array is as small as possible, but may lead to problems when decoding because it is not possible to determine the exact position of each parameter.