**Blockchain Study Notes Day 21:**

**Module 3 - Solidity Advanced**  
**Chapter 7 - Gas Fees in Solidity**

**Introduction to Gas Fees**

Gas fees are a fundamental aspect of the Ethereum network. They serve as the transaction cost users pay to interact with the blockchain, compensating miners (or validators) for their computational work in processing and validating transactions.

**1. What Are Gas Fees?**

* **Definition**:  
  Gas is a unit that measures the computational effort required to perform operations on the Ethereum blockchain.
* **Purpose**:
  + Prevents network abuse by making computation costly.
  + Incentivizes miners or validators to include transactions in blocks.

**2. Key Concepts in Gas**

**2.1. Gas Limit**

* The maximum amount of gas a user is willing to spend on a transaction.

**2.2. Gas Used**

* The actual amount of gas consumed to execute a transaction.

**2.3. Gas Price**

* The amount of Ether a user is willing to pay per unit of gas, typically specified in **Gwei**.

**2.4. Transaction Fee**

* Calculated as:

Transaction Fee = Gas Used × Gas Price

**3. Gas Costs for Common Operations**

| **Operation** | **Gas Cost** |
| --- | --- |
| Basic transaction | 21,000 gas |
| Storing a value in storage | 20,000 gas |
| Reading a value from storage | 2,100 gas |
| Adding two numbers | 3 gas |
| Calling a function | 700 gas |

**4. Example Program Demonstrating Gas (Using Munawar)**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract MunawarGasFees {

uint public storedData;

// Function to store a value in storage

function storeData(uint \_value) public {

storedData = \_value; // High gas cost due to storage write

}

// Function to read the stored value

function readData() public view returns (uint) {

return storedData; // Low gas cost due to storage read

}

// Function to perform a simple computation

function addNumbers(uint \_a, uint \_b) public pure returns (uint) {

return \_a + \_b; // Minimal gas cost for computation

}

}

**Explanation**:

1. **storeData**: Writes to storage, incurring a high gas cost.
2. **readData**: Reads from storage, incurring a lower gas cost.
3. **addNumbers**: Pure function with minimal gas usage.

**5. Gas Optimization Techniques**

**5.1. Use memory Instead of storage**

* Accessing or modifying storage is costly. Use memory for temporary variables.
* **Example**:

function processArray(uint[] memory \_array) public pure returns (uint) {

return \_array[0]; // Lower gas cost

}

**5.2. Minimize Storage Writes**

* Avoid frequent updates to storage. Batch updates where possible.

**5.3. Use calldata for External Function Parameters**

* calldata is more gas-efficient than memory for function arguments.

**5.4. Avoid Redundant Computations**

* Store results of expensive computations in variables to reuse them.

**6. Gas Fee Calculation Example**

Assume:

* **Gas Used**: 50,000
* **Gas Price**: 20 Gwei
* **1 Gwei** = 10−910^{-9}10−9 Ether

**Transaction Fee Calculation**:

Transaction Fee = Gas Used × Gas Price

= 50,000 × 20 Gwei

= 1,000,000 Gwei

= 0.001 Ether

**7. Best Practices for Managing Gas Costs**

* **Set Reasonable Gas Limits**:
  + Ensure sufficient gas to avoid failed transactions.
* **Monitor Gas Prices**:
  + Use tools like **Etherscan Gas Tracker** to determine optimal gas prices.
* **Batch Transactions**:
  + Combine multiple operations into a single transaction to save gas.

**Home Task**

1. **Extend the Example Program**:
   * Add a function to compare gas usage for different types of loops (for vs. while).
2. **Create a New Contract**:
   * Implement a contract to batch multiple storage updates in a single transaction.
3. **Research**:
   * Explore tools like **Remix** or **Hardhat** for gas profiling in Solidity.

**Conclusion**

Gas fees are an integral part of Solidity development. By understanding gas costs and implementing optimization techniques, developers can build cost-efficient and reliable smart contracts while providing a better user experience.

.

Day 21 Notes

***Prepared by Munawar Johar***