

## Лабораторна робота 1

### ОЗНАЙОМЛЕННЯ ІЗ ДЕЦЕНТРАЛІЗОВАНОЮ СИСТЕМОЮ ETHEREUM. УСТАНОВКА ТА НАЛАГОДЖЕННЯ БАЗОВОГО ПЗ

#### Теоретичні відомості

**Ethereum** (від англ. *Ether* [i:θə] – «етер», код **ETH**), **Етеріум**, (часто просто «етер» або «**ефір**») – крипто-валюта та платформа для створення децентралізованих онлайн-сервісів (dapps) на базі блокчейна, що працюють на базі розумних контрактів. Реалізована як єдина децентралізована віртуальна машина. Ідея була сформована Віталіком Бутеріном 2013 року. ETH є рідною валютою для платформи Ethereum, а також працює як плата за транзакцію майнерам у мережі Ethereum.

Центральна ідея мережі Блокчейн Ethereum підтверджена в офіційному документі Ethereum і полягає в наступному: «Мета Ethereum – створити альтернативний протокол для побудови децентралізованих додатків, надаючи інший набір компромісів, який, на нашу думку, буде дуже корисним для великого класу децентралізованих додатків з особливим акцентом на ситуації, коли важливі швидкий час розробки, безпека для невеликих і рідко використовуваних додатків і здатність різних додатків дуже ефективно взаємодіяти.

Ethereum робить це, створюючи, по суті, кінцевий абстрактний базовий рівень: Блокчейн з вбудованим повною мовою програмування Turing, що дозволяє будь-кому писати розумні контракти і децентралізовані додатки, де вони можуть створювати свої власні довільні правила для власників, форматів транзакцій і функції переходу станів».

### ***1.1 Ethereum – це програмований Блокчейн***

Ethereum – це не Біткойн-форк, і він не працює на тому ж Блокчейні, що і Біткоїни. Це абсолютно новий Блокчейн з різними технічними складнощами, але такий же за ідеєю, як і будь-який інший Блокчейн.



Рисунок 1.1 – Платформа Ethereum для створення децентралізованих онлайн-сервісів на базі блокчейна

Він заснований на тимчасовій мережі, але замість надання користувачам можливості використовувати низку визначених операцій (наприклад, криптовалютних транзакцій), як це робить Біткоїн, протокол Ethereum дозволяє користувачам виконувати практично будь-який код, який вони хочуть.

Подібно протоколу Біткоїнів, вузли в протоколі Ethereum жертвують своєю обчислювальною потужністю, щоб підтримувати і оновлювати Блокчейн. Тим не менш, вони також працюють на віртуальній машині Ethereum (EVM).

EVM – це «суперкомп'ютер», який складається з сукупної обчислювальної потужності вузлів в мережі. Цей децентралізований віртуальний суперкомп'ютер використовується для запуску інтелектуальних контрактів (код), представлених користувачами. Потім цей код зберігається в Блокчейні Ethereum і залишається там без будь-яких змін (незмінних) назавжди.

Давайте перейдемо до найдрібнішого Ефіріуму:

*Ethereum можна розглядати як такий, що має два різних компонента: облікові записи та додатки (або варіанти використання).*

## **1.2 Акаунти на Ethereum**

У протоколі Ethereum є два типи облікових записів: зовнішні облікові записи (або адреси Ethereum, контрольовані закритими ключами – як в мережі Біткойн), і контрактні облікові записи (або розумні контракти, які є адресами або «сутностями», контрольованими кодом всередині їх).

Аккаунт, що належить ззовні, використовується для створення, відправлення та отримання транзакцій і підписується закритими ключами. Уявіть, що ваш зовнішній обліковий запис схожий на банківський рахунок, а особистий ключ – це захисний код, який використовується для доступу до вашого банківського рахунку.

Як визначено в офіційному документі Ethereum, обліковий запис Ethereum містить чотири поля:

1. Одноразовий номер – лічильник, який використовується для перевірки того, що кожна транзакція може бути оброблена тільки один раз.

2. Поточний Ефірний баланс рахунку.

3. Код контракту аккаунта, якщо є (в разі смарт-контрактів).

4. Сховище облікового запису (за умовчанням порожньо).

Кожний обліковий запис Ethereum має публічну адресу або щось на зразок ряду випадкових чисел і символів в шістнадцятковому форматі, який виглядає приблизно так:

0x84dbb737eac3002103e721b9ab7ch67a6850a310

Ви можете думати про цю адресу як про «ім'я» вашого банківського рахунку, і ви будете використовувати цю адресу для надсилання та отримання Ефіру або Жетонів або використання інших сервісів по протоколу Ethereum.

Для того щоб відправити транзакцію, ви повинні підписати її своїм закритим ключем – і причина називається закритою.

Якщо хто-небудь будь-яким чином отримує ваш закритий ключ, він володіє вашим обліковим записом і може відправляти транзакції. Якщо ви втратите свій особистий ключ або парольну фразу в своєму гаманці, як це зробили багато людей, вони зникнуть назавжди. Розкіш використання децентралізованих платформ пов'язана з обов'язками, а ваша головна відповідальність - безпека вашого профілю.

### ***1.3 Установка та налаштування Ethereum***

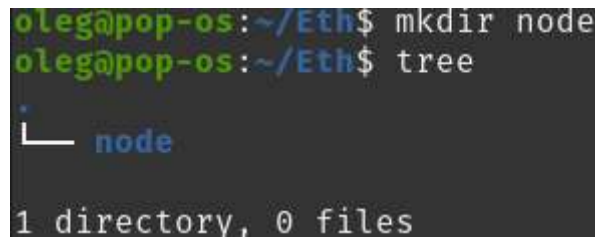
Для ознайомлення із системою Ethereum буде використовуватися офіційний клієнт – Go Ethereum (скорочено geth). Завантажити клієнт можливо і на ОС Windows, але у цьому курсі приклади будуть демонструватися на прикладі ОС Linux.

Першим кроком потрібно налаштувати Go Ethereum. Для цього потрібно по-черзі виконати відповідні команди в терміналі (рис. 1.2).

```
sudo apt-get install software-properties-common  
sudo add-apt-repository -y ppa:ethereum/ethereum  
sudo apt-get update  
sudo apt-get install ethereum
```

Рисунок 1.2 – Команди для установки клієнта Go Ethereum

За замовчуванням робоча директорія для Go Ethereum (geth надалі) – **~/.ethereum**, але для навчання її використання не є зручним, тому потрібно створити власні директорії (рис. 1.3).



```
oleg@pop-os:~/Eth$ mkdir node  
oleg@pop-os:~/Eth$ tree  
.  
└─ node  
1 directory, 0 files
```

Рисунок 1.3 – Створення потрібної структури

Після створення потрібної структури необхідно створити Ethereum акаунт (рис. 1.4). Для створення нового акаунта необхідно використати команду **account new** клієнта **geth** із опцією **--datadir <dir-name>**, яка дозволяє змінити робочу директорію за замовчуванням на **<dir-name>**.

Під час виконання команди потрібно ввести пароль від акаунта. Пароль може бути абсолютно будь-який, але треба пам'ятати, що після втрати пароля доступ до акаунта буде заблокований назавжди.

Після удачного виконання команди у вказану директорію буде додано файл із інформацією про акаунт, а саме його приватний ключ (рис. 1.5).

```
oleg@pop-os:~/Eth$ geth --datadir node/ account new
INFO [06-06|20:05:01.558] Maximum peer count          ETH=50 L1=0 total=50
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:
Your new key was generated

Public address of the key:  0xc4f043f2997521dffa9944dfc0505a2c141
Path of the secret key file: node/keystore/UTC--2021-06-06T17-05-11.502151709Z--c4f043f2997521dffa9944dfc0505a2c141

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

Рисунок 1.4 – Створення нового акаунта

```
oleg@pop-os:~/Eth$ tree
.
├── node
│   └── keystore
│       └── UTC--2021-06-06T17-05-11.502151709Z--c4f043f2997521dffa9944dfc0505a2c141
2 directories, 1 file
```

Рисунок 1.5 – Стан директорій після створення акаунта

Наступним кроком потрібно згенерувати конфіг файл для genesis блоку. Для його генерації буде використовуватися досить зручна утиліта **puppeth**.

Після запуску **puppeth** потрібно вибрати назву для нової мережі. Потім необхідно вибрати потрібну конфігурацію за допомогою діалогового режиму в терміналі (рис. 1.6).

```
oleg@pop-os:~/eth$ puppeth
-----
| Welcome to puppeth, your Ethereum private network manager |
| This tool lets you create a new Ethereum network down to |
| the genesis block, bootnodes, miners and ethstats servers |
| without the hassle that it would normally entail.         |
| Puppeth uses SSH to dial in to remote servers, and builds |
| its network components out of Docker containers using the |
| docker-compose toolset.                                   |
-----

Please specify a network name to administer (no spaces, hyphens or capital letters please)
> testnet

Sweet, you can set this via --network-testnet next time!

INFO [06-06|20:07:13.850] Administering Ethereum network      name=testnet
WARN [06-06|20:07:13.850] No previous configurations found    path=/home/oleg/.puppeth/testnet

What would you like to do? (default = stats)
  1. Show network stats
  2. Configure new genesis
  3. Track new remote server
  4. Deploy network components
> 2

What would you like to do? (default = create)
  1. Create new genesis from scratch
  2. Import already existing genesis
> 1

Which consensus engine to use? (default = clique)
  1. Ethash - proof-of-work
  2. Clique - proof-of-authority
> 1

Which accounts should be pre-funded? (advisable at least one)
> 0xc4f043f2997521dFF214EFa9944dFC0505A2C141
> 0x

Should the precompile-addresses (0x1 .. 0xff) be pre-funded with 1 wei? (advisable yes)
> n

Specify your chain/network ID if you want an explicit one (default = random)
> 9988
INFO [06-06|20:08:41.025] Configured new genesis block
```

Рисунок 1.6 – Створення та конфігурація genesis.json

Щоб застосувати створену конфігурацію потрібно експортувати її у файли із розширенням .json (рис. 1.7).

```
What would you like to do? (default = stats)
1. Show network stats
2. Manage existing genesis
3. Track new remote server
4. Deploy network components
> 2

1. Modify existing configurations
2. Export genesis configurations
3. Remove genesis configuration
> 2

Which folder to save the genesis specs into? (default = current)
Will create testnet.json, testnet-aleth.json, testnet-harmony.json, testnet-parity.json
>

INFO [06-06|20:09:08.319] Saved native genesis chain spec      path=testnet.json
INFO [06-06|20:09:08.321] Saved genesis chain spec             client=aleth path=testnet-aleth.json
INFO [06-06|20:09:08.325] Saved genesis chain spec             client=parity path=testnet-parity.json
INFO [06-06|20:09:08.325] Saved genesis chain spec             client=harmony path=testnet-harmony.json
```

Рисунок 1.7 – Експорт сконфігурованого genesis.json файлу

Нам буде потрібен лише <your-network-name>.json, тому всі інші можна видалити після експорту (рис. 1.8).

```
oleg@pop-os:~/Eth$ tree
.
├── node
│   └── keystore
│       └── UTC--2021-06-06T17-05-11.502151709Z--c4f043f2997521dff214efa9944dfc0505a2c141
├── testnet-aleth.json
├── testnet-harmony.json
├── testnet.json
└── testnet-parity.json

2 directories, 5 files
```

Рисунок 1.8 – Стан директорій після експорту

Сам файл конфігурації включає в себе набір полів у форматі .json (рис. 1.9). У ньому також указуються адреса акаунтів, на яких одразу буде нараховано велику кількість ЕТН. Ці адреса вказуються для поля alloc разом із кількістю ЕТН, який буде нараховано.



```

{
  "config": {
    "chainId": 9988,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "ethash": {}
  },
  "nonce": "0x0",
  "timestamp": "0x60bd00d9",
  "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "gasLimit": "0x47b760",
  "difficulty": "0x80000",
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "alloc": {
    "c4f043f2997521dffa214efa9944dfc0505a2c141": {
      "balance": "0x2000000000000000000000000000000000000000000000000000000000000000"
    }
  },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000"
}

```

Рисунок 1.9 – Вміст експортованого конфігу

Тепер потрібно ініціалізувати нашу вибрану директорію створеною конфігурацією (рис. 1.10).

```

$ ./geth --datadir node/ init testnet.json
[2018-06-06T11:51:44Z] Maximum peer count
[2018-06-06T11:51:44Z] Set global gas cap
[2018-06-06T11:51:44Z] Allocated cache and file handles
[2018-06-06T11:51:44Z] Writing custom genesis block
[2018-06-06T11:51:47Z] Persisted trie from memory database
[2018-06-06T11:51:47Z] Successfully wrote genesis state
[2018-06-06T11:51:47Z] Allocated cache and file handles
[2018-06-06T11:51:47Z] Writing custom genesis block
[2018-06-06T11:51:47Z] Persisted trie from memory database
[2018-06-06T11:51:47Z] Successfully wrote genesis state

ethash=50 (1x=0 (cpu)=50
cpu=25,000,000
balance=/home/oleg/eth/node/geth/chaindata 0x0=10.00Mio 0x0000000000000000000000000000000000000000000000000000000000000000
nodes=1 1170=173.88M 100="70.147ps" 100000=8 100000=0.000 100000=0.000 100000=1 100000=0.000
chaindata=chaindata 100000=0.000 100000=0.000 100000=0.000 100000=0.000 100000=0.000 100000=0.000 100000=0.000 100000=0.000
balance=/home/oleg/eth/node/geth/lightchaindata 100000=10.00Mio 100000=0.000
nodes=1 1170=173.88M 1000="46.479ps" 100000=8 100000=0.000 100000=0.000 100000=1 100000=0.000
chaindata=lightchaindata 100000=0.000 100000=0.000 100000=0.000 100000=0.000 100000=0.000 100000=0.000 100000=0.000 100000=0.000

```

Рисунок 1.10 – Ініціалізація ноди genesis блоку створеною конфігурацією

Для цього потрібно виконати команду `init <your-network-name>.json`. Ініціалізувати директорію потрібно лише один раз. Після успішної ініціалізації директорії в ній повинні з'явитися системні файли (рис. 1.11).

```
oleg@pop-os:~/Eth$ tree
.
├── node
│   ├── geth
│   │   ├── chaindata
│   │   │   ├── 000001.log
│   │   │   ├── CURRENT
│   │   │   ├── LOCK
│   │   │   ├── LOG
│   │   │   └── MANIFEST-000000
│   │   ├── lightchaindata
│   │   │   ├── 000001.log
│   │   │   ├── CURRENT
│   │   │   ├── LOCK
│   │   │   ├── LOG
│   │   │   └── MANIFEST-000000
│   │   ├── LOCK
│   │   └── nodekey
│   └── keystore
│       └── UTC--2021-06-06T17-05-11.502151709Z--c4f043f2997521dff214efa9944dfc0505a2c141
└── testnet.json
```

Рисунок 1.11– Стан директорій після ініціації genesis блоку

Тепер в нас є налаштовано до запуску ноди директорія (рис. 1.11). Для запуску ноди існує досить багато різних опцій, ось основні з них:

- `--syncmode` - вказує тип синхронізації ноди
- `--rpc` - вмикає HTTP-RPC сервер
- `--rpcaddr` - вказує адрес HTTP-RPC сервера
- `--rpcport` - вказує порт HTTP-RPC сервера
- `--rpcapi` - вказує API, які будуть доступні через HTTP-RPC сервер
- `--networkid` - вказує на ID мережі, який був вказаний в

конфігураційному файлі для genesis блоку

Після успішного запуску в директорії повинен з'явитися `geth.ipc`, який потрібен для ввімкнення консолі, та інші файли (рис. 1.12).

[illegible]

Рисунок 1.12 – Запуск повної ноди

Для взаємодії із нодами існують JavaScript console. Ми можемо підключитися до ввімкненої ноди використовуючи команду **attach** <path-to-geth.ipc> (рис. 1.13).

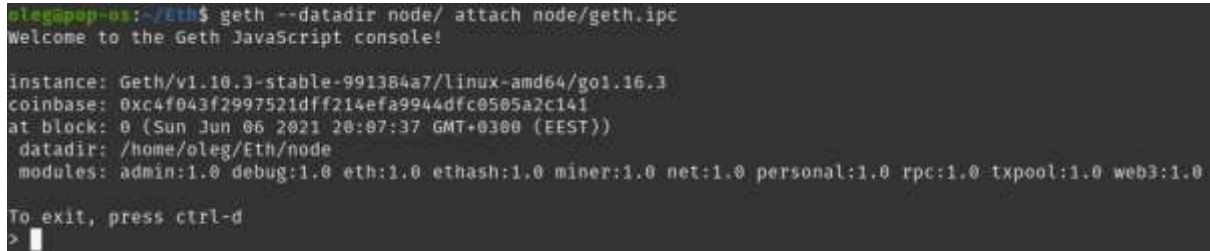
```

blegaptop-os:~/eth$ tree
.
├── node
│   ├── geth
│   │   ├── chaindata
│   │   │   ├── 000002.ldb
│   │   │   ├── 000003.log
│   │   │   ├── ancient
│   │   │   │   ├── bodies.0000.cdat
│   │   │   │   ├── bodies.cidx
│   │   │   │   ├── diffs.0000.rdat
│   │   │   │   ├── diffs.ridx
│   │   │   │   ├── FLOCK
│   │   │   │   ├── hashes.0000.rdat
│   │   │   │   ├── hashes.ridx
│   │   │   │   ├── headers.0000.cdat
│   │   │   │   ├── headers.cidx
│   │   │   │   ├── receipts.0000.cdat
│   │   │   │   └── receipts.cidx
│   │   │   ├── CURRENT
│   │   │   ├── CURRENT.bak
│   │   │   ├── LOCK
│   │   │   ├── LOG
│   │   │   └── MANIFEST-000004
│   │   ├── lightchaindata
│   │   │   ├── 000001.log
│   │   │   ├── CURRENT
│   │   │   ├── LOCK
│   │   │   ├── LOG
│   │   │   └── MANIFEST-000000
│   │   ├── LOCK
│   │   ├── nodekey
│   │   ├── nodes
│   │   │   ├── 000001.log
│   │   │   ├── CURRENT
│   │   │   ├── LOCK
│   │   │   ├── LOG
│   │   │   └── MANIFEST-000000
│   │   ├── transactions.rlp
│   │   ├── geth.ipc
│   │   ├── keystore
│   │   │   └── UTC--2021-06-06T17-05-11.502151709Z---c4f043f2997521dff214efa9944dfc0505a2c141
│   └── testnet.json
└── 7 directories, 34 files

```

Рисунок 1.13 – Стан директорій після запуску ноди

Також є можливість просто ввімкнути консоль за допомогою команди **console** (рис. 1.14).



```
oleg@pop-os: ~/Eth$ geth --datadir node/ attach node/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.3-stable-991384a7/linux-amd64/go1.16.3
coinbase: 0xc4f043f2997521dffa214efa9944dfc0505a2c141
at block: 0 (Sun Jun 06 2021 20:07:37 GMT+0300 (EEST))
datadir: /home/oleg/Eth/node
modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d
> 
```

Рисунок 1.14 – Підключення до консолі ноди

Команди для ознайомлення з консоллю:

- `eth.accounts` – виводить список акаунтів;
- `eth.getBalance` – повертає баланс акаунта в ЕТН;
- `personal.newAccount` – створює новий акаунт.

### Індивідуальне завдання

1. Вивести список всіх акаунтів.
2. Перевірити баланс існуючих акаунтів.
3. Створити новий акаунт.
4. Перевірити баланс нового акаунта.
5. Звіт виконаної роботи.