FinMiner by FINOM
version: 2.1

FinMiner is a program product developed by the Finom company to create structural cryptocurrency units on the framework of the Ethash and CryptoNight algorithms. This present version of **FinMiner** was made to work with every cryptocurrency based on those algorithms: Ethereum, Ethereum Classic, Monero, Electroneum, etc. This version of **FinMiner** was developed to run on Windows and Linux OS with Nvidia graphics cards.

Testing on **FinMiner** demonstrated high effectiveness working with Ethereum, Ethereum Classic, Electroneum and other currencies. As a result of the research carried out, it was found that **FinMiner** performs on par with, and sometimes better than, competing program products. Regardless, **FinMiner** stands apart with its high stability and simple setup.

Payment

Payment for the use of **FinMiner** takes the form of a commission from mining. The commission is 1% of total mining time: every hour at random **FinMiner** switches the mining to its wallets for 36 seconds.

Setup

At launch **FinMiner** reads the *config.ini* setup file from the program’s current directory. In order to assign a specific name to the config file, it should be written as the first argument in the command line. For example:

```
finminer.exe config_etc.ini
```

When using **FinMiner** it is not necessary to specify the pools in the config file. If the pool (list of pools) is not specified, **FinMiner** will automatically use the pools on nanopool.org in accordance with the chosen cryptocurrency.

When **FinMiner** starts up it displays in the console log the main work information, including the program’s current version, name of the rig, number and type of graphics cards installed, and the program’s current configurations.

Log Files

When it is running **FinMiner** creates log files and saves them in the logs folder of the program’s current directory.

Parameters

The configurations of **FinMiner** are in the configuration file with the *.ini extension (*config.ini* by default) in the form of “parameter=definition” logic pairs. This file permits the presence of empty lines and comments. Comment lines should begin with a **;** (semicolon). Parameters and definitions are not case-sensitive, which means it makes no difference to the program whether you type ETH, eth or Eth.

Here is a list of parameters that can be set in **FinMiner**:

wallet

Mandatory parameter — the user’s wallet, into which funds will be deposited.

paymentId

Optional parameter, can be defined for wallets created on an exchange where the user has a personal payment number in addition to their wallet.

algorithm

Optional parameter, which can be defined as “Ethash” or “Cryptonight”. If this parameter is not specified, **FinMiner** will determine the algorithm on the basis of the cryptocurrency used or the format of the wallet specified.

coin

Optional parameter, can be defined as ETH, ETC, XMR and ETN for Ethereum, Ethereum Classic, Monero and Electroneum, respectively. When a coin is specified **FinMiner** automatically determines the algorithm and necessary pools for it to function (on nanopool.org) if none are provided in a separate parameter.

rigname

Optional parameter, consisting of the name of the rig (computer/other hardware). Displayed in the pool’s statistics. If this parameter is not defined, the program will generate a unique rig name for the rig and provide it to the pool.

email

Optional parameter, consisting of the user’s e-mail address. It is provided to the pool where the rig will be operating. The pool can use it when sending out service notifications.

pool1, pool2, …

Optional parameter, consisting of mining pools. Definitions must be given in the format url:port (for example: “pool1=eth-eu1.nanopool.org:9999”). The parameters should be defined in ascending, sequential order, from pool1 to poolN (for example: pool1, pool2, pool3).

Configuration File

In order to begin working with **FinMiner**, it’s enough to simply input your wallet in the configuration file.

For example, to earn Ethereum the configuration file should look like so:

```
wallet=<user’s wallet>
```

FinMiner will automatically use Ethereum pools.

In order to work with Ethereum Classic, the coin must be specified:

```
wallet=<user’s wallet>
coin=ETC
```

In this case **FinMiner** will use pools corresponding to Ethereum Classic.

Similarly, in order to mine Monero or Electroneum it is necessary to specify a wallet and payment ID (if available). If the wallet is 95 characters or more, then **FinMiner** will automatically determine that it is a wallet for the CryptoNight algorithm. If an Electroneum wallet begins with “etn,” **FinMiner** will automatically determine the algorithm and pool without any need to specify the coin.

IMPORTANT!

For coins that run on the Ethash and CryptoNight algorithms but which are not supported by nanopool.org, **you must** specify a **wallet**, **paymentId**, **algorithm** and pool (**pool1…**). In this case **FinMiner** will function as if it were working with Ethereum or Monero but the results of its operations will be determined by tasks from the corresponding pools specified in the configuration file.

Examples of Configuration Files

Example of complete configuration file for Ethereum:

```
wallet = 0xffffffffffffffffffffffffffffffffffffffff
algorithm = Ethash
rigName = rig1
email = someemail@org
pool1 = eth-eu1.nanopool.org:9999
pool2 = eth-eu2.nanopool.org:9999
pool3 = eth-us-east1.nanopool.org:9999
pool4 = eth-us-west1.nanopool.org:9999
pool5 = eth-asia1.nanopool.org:9999
```

Example of an equivalent file for Ethereum:

```
wallet = 0xffffffffffffffffffffffffffffffffffffffff
rigName = rig1
email = someemail@org
```

Example of a minimum file for Ethereum:

```
wallet=0xffffffffffffffffffffffffffffffffffffffff
```

Example of a complete file for Ethereum Classic:

```
wallet = 0xffffffffffffffffffffffffffffffffffffffff
algorithm = Ethash
coin=Etc
rigName = rig1
email = someemail@org
pool1 = etc-eu1.nanopool.org:19999
pool2 = etc-eu2.nanopool.org:19999
pool3 = etc-us-east1.nanopool.org:19999
pool4 = etc-us-west1.nanopool.org:19999
pool5 = etc-asia1.nanopool.org:19999
```

Example of an equivalent file for Ethereum Classic:

```
wallet = 0xffffffffffffffffffffffffffffffffffffffff
coin=Etc
rigName = rig1
email = someemail@org
```

Example of a minimum file for Ethereum Classic:

```
wallet=0xffffffffffffffffffffffffffffffffffffffff
coin=Etc
```

Example of a complete file for Monero:

```
wallet = XXXXXXXXXX
paymentId=YYYYYYYYYY
algorithm = Cryptonight
rigName = rig1
email = someemail@org
pool1 = xmr-eu1.nanopool.org:14433
pool2 = xmr-eu2.nanopool.org:14433
pool3 = xmr-us-east1.nanopool.org:14433
pool4 = xmr-us-west1.nanopool.org:14433
pool5 = xmr-asia1.nanopool.org:14433
```

Example of an equivalent file for Monero:

```
wallet = XXXXXXXXXX
paymentId=YYYYYYYYYY
rigName = rig1
email = someemail@org
```

Example of a minimum file for Monero:

```
wallet = XXXXXXXXXX
```

Example of a complete file for Electroneum:

```
wallet = etnXXXXXXXXXX
paymentId = YYYYYYYYYY
algorithm = Cryptonight
coin = ETN
rigName = rig1
email = someemail@org
pool1 = etn-eu1.nanopool.org:13433
pool2 = etn-eu2.nanopool.org:13433
pool3 = etn-us-east1.nanopool.org:13433
pool4 = etn-us-west1.nanopool.org:13433
pool5 = etn-asia1.nanopool.org:13433
```

Example of an equivalent file for Electroneum:

```
wallet = etnXXXXXXXXXX
paymentId = YYYYYYYYYY
coin = ETN
```

```
rigName = rig1
email = someemail@org
```

Example of a minimum file for Electroneum:

```
wallet = etnXXXXXXXXXXX
```