FinMiner de FINOM
versión 2.1

FinMiner es un programa-producto desarrollado por la compañía Finom para crear unidades estructurales de criptomoneda que utilizan los algoritmos Ethash y Cryptonight. La versión actual de **FinMiner** fue creada para manejar todas las criptomonedas basadas en esos algoritmos: Ethereum, Ethereum Classic, Monero, Electroneum, etc. Esta versión de **FinMiner** puede ser usada con Windows o Linux y con tarjetas de video Nvidia.

Las pruebas que se han hecho con **FinMiner** muestran un alto rendimiento al trabajar con Ethereum, Ethereum Classic, Electroneum y otras monedas. Los resultados de la investigación demuestran que **FinMiner** trabaja a un nivel parejo o mayor que el de la competencia. **FinMiner** también se distingue por su alta estabilidad y simple configuración.

Pago

Pago por el uso de **FinMiner** toma la forma de una comisión de la minería. La comisión es 1% del tiempo total de minería: una vez cada hora al azar **FinMiner** desvía la minería a sus monederos por 36 segundos.

Configuración

Al abrirse, **FinMiner** lee el archivo config.ini del directorio actual del programa. Para asignarle un nombre específico al archivo de configuración, debería de ser escrito como el primer argumento en la línea de comandos. Por ejemplo:

finminer.exe config_etc.ini

Al usar **FinMiner**, no es necesario especificar los pools en el archivo de configuración. Si el pool (lista de pools) no se especifica, **FinMiner** automáticamente utilizará los pools en nanopool.org para la criptomoneda escogida.

FinMiner al abrir muestra en el log (registro) de la consola la información principal de trabajo, que incluye la versión actual del programa, el nombre del rig, la cantidad y tipo de tarjetas de video instaladas y las configuraciones actuales del programa.

Archivos log

Cuando está activado, **FinMiner** crea archivos log y los guarda en el folder de logs del directorio actual del programa.

Parámetros

La configuración de **FinMiner** se realiza en el archivo de configuración con la extensión *.ini (config.ini por defecto) en la forma de pares de lógica parámetro=definición. Este archivo permite la presencia de líneas y comentarios vacíos. Las líneas de comentario deben de comenzar con un ; (punto y coma). Los parámetros y definiciones no son sensibles a las mayúsculas, lo cual significa que para el programa no hay ninguna diferencia si escribes ETH, eth o Eth.

Aquí hay una lista de los parámetros que pueden ser configurados en **FinMiner**:

wallet

Parámetro mandatorio - el monedero del usuario, a donde se depositarán fondos.

paymentId

Parámetro opcional, puede ser definido para monederos creados en una casa de cambio donde el usuario tiene un número personal de pago aparte de su monedero.

algorithm

Parámetro opcional, puede ser definido como “Ethash” o “Cryptonight”. Si este parámetro no se especifica, **FinMiner** determinará el algoritmo a base de la criptomoneda usada o el formato del monedero especificado.

coin

Parámetro opcional, puede ser definido como ETH, ETC, XMR y ETN para Ethereum, Ethereum Classic, Monero y Electroneum, respectivamente. Cuando se especifica una moneda, **FinMiner** automáticamente determina el algoritmo y pools necesarios para que funcione (en nanopool.org) si no se especifican en un parámetro separado.

rigname

Parámetro opcional, consiste en el nombre del rig (computadora o equipo). Muestra las estadísticas del pool. Si este parámetro no se defina, el programa generará un nombre único para el rig y lo pasará al pool.

email

Parámetro opcional, consiste en la dirección de correo electrónico del usuario. Se proporciona al pool donde estará operando el rig. El pool lo puede usar para mandar notificaciones de servicio.

pool1, pool2, …

Parámetro opcional, consiste en pools de minería. Las definiciones deben de estar en el formato url:port (por ejemplo: “pool1=eth-eu1.nanopool.org:9999”). Los parámetros deben ser definidos en orden ascendente y secuencial, de pool1 a poolN (por ejemplo: pool1, pool2, pool3).

Archivo de configuración

Para poder empezar a trabajar con **FinMiner**, basta sólo con ingresar tu número de monedero en el archivo de configuración.

Por ejemplo, para ganar Ethereum el archivo de configuración debe de ser así:

```
wallet=<user's wallet>
```

FinMiner automáticamente utilizará pools para Ethereum.

Para trabajar con Ethereum Classic, la moneda debe de ser especificada:

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

En este caso **FinMiner** utilizará los pools que le corresponden a Ethereum Classic.

De manera parecida, para minar Monero o Electroneum es necesario especificar un monedero y un número personal de identificación (en caso de tener uno). Si el monedero es de 95 símbolos o más, **FinMiner** automáticamente determinará que es un monedero para el algoritmo CryptoNight. Si un monedero de Electroneum empieza con "etn", **FinMiner** automáticamente determinará el algoritmo y pool sin necesidad de especificar la moneda.

¡IMPORTANTE!

Para monedas que funcionan con los algoritmos Ethash y CryptoNight pero que no están apoyados por nanopool.org, **tienes que** especificar **wallet**, **paymentId**, algorithm y pool (**pool1**…). En este caso **FinMiner** funcionará como si estuviera trabajando con Ethereum o Monero, pero los resultados de las operaciones se determinarán por los pools en el archivo de configuración.

Ejemplos de archivos de configuración

Un ejemplo de un archivo de configuración completo para 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
```

Un ejemplo de un archivo equivalente para Ethereum:

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

Un ejemplo de un archivo mínimo para Ethereum:

```
wallet=0xffffffffffffffffffffffffffffffffffffffff
```

Un ejemplo de un archivo completo para 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
```

Un ejemplo de un archivo equivalente para Ethereum Classic:

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

Un ejemplo de un archivo mínimo para Ethereum Classic:

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

Un ejemplo de un archivo completo para 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
```

Un ejemplo de un archivo equivalente para Monero:

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

Un ejemplo de un archivo mínimo para Monero:

```
wallet = XXXXXXXXXX
```

Un ejemplo de un archivo completo para 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
```

Un ejemplo de un archivo equivalente para Electroneum:

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

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

Un ejemplo de un archivo mínimo para Electroneum:

```
wallet = etnXXXXXXXXXX
```