以太坊的目的是基于脚本、竞争币和链上元协议(on-chain meta-protocol)概念进行整合和提高，使得开发者能够创建任意的基于共识的、可扩展的、标准化的、特性完备的、易于开发的和协同的应用。以太坊通过建立终极的抽象的基础层-内置有图灵完备编程语言的区块链-使得任何人都能够创建合约和去中心化应用并在其中设立他们自由定义的所有权规则、交易方式和状态转换函数。域名币的主体框架只需要两行代码就可以实现，诸如货币和信誉系统等其它协议只需要不到二十行代码就可以实现。智能合约-包含价值而且只有满足某些条件才能打开的加密箱子-也能在我们的平台上创建，并且因为图灵完备性、价值知晓(value-awareness)、区块链知晓(blockchain-awareness)和多状态所增加的力量而比比特币脚本所能提供的智能合约强大得多。以太坊账户在以太坊系统中，状态是由被称为“账户”(每个账户由一个20字节的地址)的对象和在两个账户之间转移价值和信息的状态转换构成的。以太坊的账户包含四个部分：随机数，用于确定每笔交易只能被处理一次的计数器账户目前的以太币余额账户的合约代码，如果有的话账户的存储(默认为空)以太币(Ether)是以太坊内部的主要加密燃料，用于支付交易费用。一般而言，以太坊有两种类型的账户：外部所有的账户(由私钥控制的)和合约账户(由合约代码控制)。外部所有的账户没有代码，人们可以通过创建和签名一笔交易从一个外部账户发送消息。每当合约账户收到一条消息，合约内部的代码就会被激活，允许它对内部存储进行读取和写入，和发送其它消息或者创建合约。消息和交易以太坊的消息在某种程度上类似于比特币的交易，但是两者之间存在三点重要的不同。第一，以太坊的消息可以由外部实体或者合约创建，然而比特币的交易只能从外部创建。第二，以太坊消息可以选择包含数据。第三，如果以太坊消息的接受者是合约账户，可以选择进行回应，这意味着以太坊消息也包含函数概念。以太坊中“交易”是指存储从外部账户发出的消息的签名数据包。交易包含消息的接收者、用于确认发送者的签名、以太币账户余额、要发送的数据和两个被称为STARTGAS和GASPRICE的数值。为了防止代码的指数型爆炸和无限循环，每笔交易需要对执行代码所引发的计算步骤-包括初始消息和所有执行中引发的消息-做出限制。STARTGAS就是限制，GASPRICE是每一计算步骤需要支付矿工的费用。如果执行交易的过程中，“用完了瓦斯”，所有的状态改变恢复原状态，但是已经支付的交易费用不可收回了。如果执行交易中止时还剩余瓦斯，那么这些瓦斯将退还给发送者。创建合约有单独的交易类型和相应的消息类型;合约的地址是基于账号随机数和交易数据的哈希计算出来的。消息机制的一个重要后果是以太坊的“头等公民”财产-合约与外部账户拥有同样权利，包括发送消息和创建其它合约的权利。这使得合约可以同时充当多个不同的角色，例如，用户可以使去中心化组织(一个合约)的一个成员成为一个中介账户(另一个合约)，为一个偏执的使用定制的基于量子证明的兰波特签名(第三个合约)的个人和一个自身使用由五个私钥保证安全的账户(第四个合约)的共同签名实体提供居间服务。以太坊平台的强大之处在于去中心化的组织和代理合约不需要关心合约的每一参与方是什么类型的账户。以太坊状态转换函数

以太坊的状态转换函数：APPLY(S,TX) ->

S'，可以定义如下：检查交易的格式是否正确(即有正确数值)、签名是否有效和随机数是否与发送者账户的随机数匹配。如否，返回错误。计算交易费用:fee=STARTGAS * GASPRICE，并从签名中确定发送者的地址。从发送者的账户中减去交易费用和增加发送者的随机数。如果账户余额不足，返回错误。设定初值GAS = STARTGAS，并根据交易中的字节数减去一定量的瓦斯值。从发送者的账户转移价值到接收者账户。如果接收账户还不存在，创建此账户。如果接收账户是一个合约，运行合约的代码，直到代码运行结束或者瓦斯用完。如果因为发送者账户没有足够的钱或者代码执行耗尽瓦斯导致价值转移失败，恢复原来的状态，但是还需要支付交易费用，交易费用加至矿工账户。否则，将所有剩余的瓦斯归还给发送者，消耗掉的瓦斯作为交易费用发送给矿工。 例如，假设合约的代码如下：if !self.storage[calldataload(0)]: self.storage[calldataload(0)] = calldataload(32) 需要注意的是，在现实中合约代码是用底层以太坊虚拟机(EVM)代码写成的。上面的合约是用我们的高级语言Serpent语言写成的，它可以被编译成EVM代码。假设合约存储器开始时是空的，一个值为10以太，瓦斯为2000，瓦斯价格为0.001以太并且64字节数据，第一个三十二字节的快代表号码2和第二个代表词CHARLIE。的交易发送后，状态转换函数的处理过程如下：检查交易是否有效、格式是否正确。检查交易发送者至少有2000*0.001=2个以太币。如果有，从发送者账户中减去2个以太币。初始设定gas=2000,假设交易长为170字节，每字节的费用是5，减去850，所以还剩1150。从发送者账户减去10个以太币，为合约账户增加10个以太币。运行代码。在这个合约中，运行代码很简单：它检查合约存储器索引为2处是否已使用，注意到它未被使用，然后将其值置为CHARLIE。假设这消耗了187单位的瓦斯，于是剩余的瓦斯为1150 - 187 = 963。 6.
向发送者的账户增加963*0.001=0.963个以太币，返回最终状态。 如果没有合约接收交易，那么所有的交易费用就等于GASPRICE乘以交易的字节长度，交易的数据就与交易费用无关了。另外，需要注意的是，合约发起的消息可以对它们产生的计算分配瓦斯限额，如果子计算的瓦斯用完了，它只恢复到消息发出时的状态。因此，就像交易一样，合约也可以通过对它产生的子计算设置严格的限制，保护它们的计算资源。代码执行以太坊合约的代码使用低级的基于堆栈的字节码的语言写成的，被称为“以太坊虚拟机代码”或者“EVM代码”。代码由一系列字节构成，每一个字节代表一种操作。一般而言，代码执行是无限循环，程序计数器每增加一(初始值为零)就执行一次操作，直到代码执行完毕或者遇到错误，STOP或者RETURN指令。操作可以访问三种存储数据的空间：堆栈，一种后进先出的数据存储，32字节的数值可以入栈，出栈。内存，可无限扩展的字节队列。合约的长期存储，一个秘钥/数值的存储，其中秘钥和数值都是32字节大小，与计算结束即重置的堆栈和内存不同，存储内容将长期保持。代码可以象访问区块头数据一样访问数值，发送者和接受到的消息中的数据，代码还可以返回数据的字节队列作为输出。EVM代码的正式执行模型令人惊讶地简单。当以太坊虚拟机运行时，它的完整的计算状态可以由元组(block_state, transaction, message, code, memory, stack, pc, gas)来定义，这里block_state是包含所有账户余额和存储的全局状态。每轮执行时，通过调

出代码的第pc(程序计数器)个字节，当前指令被找到，每个指令都有定义自己如何影响元组。例如，ADD将两个元素出栈并将它们的和入栈，将gas(瓦斯)减一并将pc加一，SSTORE将顶部的两个元素出栈并将第二个元素插入到由第一个元素定义的合约存储位置，同样减少最多200的gas值并将pc加一，虽然有许多方法通过即时编译去优化以太坊，但以太坊的基础性的实施可以用几百行代码实现。区块链和挖矿

虽然有一些不同，但以太坊的区块链在很多方面类似于比特币区块链。它们的区块链架构的不同在于，以太坊区块不仅包含交易记录和最近的状态，还包含区块序号和难度值。以太坊中的区块确认算法如下：检查区块引用的上一个区块是否存在和有效。检查区块的时间戳是否比引用的上一个区块大，而且小于15分钟。检查区块序号、难度值、 交易根，叔根和瓦斯限额(许多以太坊特有的底层概念)是否有效。检查区块的工作量证明是否有效。将S[0]赋值为上一个区块的STATE_ROOT。将TX赋值为区块的交易列表，一共有n笔交易。对于属于0.....n-1的i，进行状态转换S[i+1] = APPLY(S[i],TX[i])。如果任何一个转换发生错误，或者程序执行到此处所花费的瓦斯(gas)超过了GASLIMIT，返回错误。用S[n]给S_FINAL赋值,
向矿工支付区块奖励。 8 检查S-FINAL是否与STATE_ROOT相同。如果相同，区块是有效的。否则，区块是无效的。这一确认方法乍看起来似乎效率很低，因为它需要存储每个区块的所有状态，但是事实上以太坊的确认效率可以与比特币相提并论。原因是状态存储在树结构中(tree structure)，每增加一个区块只需要改变树结构的一小部分。因此，一般而言，两个相邻的区块的树结构的大部分应该是相同的，因此存储一次数据，可以利用指针(即子树哈希)引用两次。一种被称为“帕特里夏树”（“Patricia Tree”)的树结构可以实现这一点，其中包括了对默克尔树概念的修改，不仅允许改变节点，而且还可以插入和删除节点。另外，因为所有的状态信息是最后一个区块的一部分，所以没有必要存储全部的区块历史-这一方法如果能够可以应用到比特币系统中，经计算可以对存储空间有10-20倍的节省。ETH官网：https://www.ethereum.org/ETH钱包下载：https://geth.ethereum.org/downloads/交易平台：http://www.qukuaiwang.com.cn/pingtai.html(找适合自己的)矿池：https://www.bw.com/https://www.f2pool.com/https://www.antpool.com/