万物皆有编号

从快递单号到经纬度：一套好编号是怎样设计出来的

费曼式写法 · 由多个 AI 代理撰写与互相审校

导读

从快递单号到经纬度：一套好编号是怎样设计出来的

为什么读这本书

先给你看一个有点浪费的东西：你的快递单号有十五位。十五位数字意味着一千万亿种组合，而全中国一天的快递量大约三亿多件。**盒子比东西大几百万倍。**是快递公司数学不好吗？恰恰相反——多出来的每一位，都是被现实狠狠教训过一次之后加上去的：有的是为了扛住双十一，有的是为了不让新包裹和三个月前的旧包裹撞号，还有的纯粹是给还没发明的业务留个座。

这本书就是从这根「过长」的单号出发的。它想回答一个你大概从没认真想过、却天天在用的问题：**世界为什么需要给一切编号？一套好的编号，又长什么样？**

> 大白话
>
> 编号这东西，平时你感觉不到它存在——就像你感觉不到空气。可一旦它失灵，世界立刻乱套：重名的人拿错药，撞号的包裹寄错城，抄错一位的银行卡号把钱送进陌生人口袋。编号的本质，是把「那一个」从「那一堆」里无条件地挑出来，不靠形容，不靠运气。

我们不会罗列知识点，而是带着问题去读：为什么名字在大规模系统里必然失灵？为什么有的编号给人看、有的给机器看，两者天生打架？为什么银行卡号最后一位能抓住你九成的手滑？为什么号码一旦发出去，就再也收不回来了？每一章，我们都拆开一个你见过的编号——航班号、二维码、身份证号、经纬度——看它的每一位是怎么被设计出来的，又为什么是今天这副模样。

读完这本书，你再看到任何一串数字，眼光会不一样：你会忍不住去猜，这一段是什么，那一位在防什么，设计它的人当年踩过什么坑。好了，从那根十五位的绳子开始拆吧。

第一章 · 快递单号为什么那么长

先别把它当一串乱码。找一张手边的快递面单，盯着那串数字看十秒钟——比如京东物流的单号，十五位，以 `JD` 开头。十五位纯数字意味着一千万亿种组合，而中国全国一天的快递量是多少？根据国家邮政局的公报，2023 年全国快递业务量 1320.7 亿件，摊到每天大约 3.6 亿件。

也就是说，快递公司给自己准备了一个能装一千万亿的盒子，每天只往里扔 3.6 亿个包裹。**容量比需求大了几百万倍。**是这些公司数学不好吗？恰恰相反——多出来的每一位，都是被现实教训过之后加上去的。这一章我们就把这根十五位的绳子拆开，看看每一股里藏着什么。

编号不是字符串，是压缩包

拆开一根典型的快递单号，你会发现它大致是几段拼起来的：公司标识、日期、网点代码——揽收你这个包裹的那个营业点在全网的几千个网点里排第几——再加上一段流水号，就是当天这个网点经手的第几个件。

> 大白话
>
> 所谓「编码」，说穿了就一件事：把本来要用一句话才能说清的信息，折叠进一串固定长度的字符里。就像写信封，「北京市海淀区中关村大街 27 号」是一句话，邮编 100080 是它的压缩版。快递单号干的也是这个，只不过它压缩的不只是地点，还有时间和经手人。

为什么要这么设计？想想分拣中心那一幕：传送带上的包裹一秒过一个，扫码枪「嘀」一声，系统要在几十毫秒内回答三个问题——这单是谁收的、哪天收的、接下来该往哪条流水线走。如果单号只是数据库里一个毫无含义的递增序号，那每回答一次就得查一次中央数据库；而把网点和日期直接编进号码里，分拣机在本地就能把第一段拆出来，当场做路由判断。**号码本身成了随身携带的档案袋。**

这就是这套编号教我们的第一件事：好的编号不是随机字符串，而是一套压缩过的信息设计——每一位都有岗位，每一位都在替系统省一次查询、省一次追问。

位数是被谁逼出来的

那为什么要十五位？十位，一百亿，也够装一年的量了啊。

问题出在「流水号从哪天开始归零」上。假如单号只有九位，一个繁忙网点一天收十万件，八位就烧完了；更要命的是号码会循环复用——三个月前那个已经签收的包裹的单号，今天又发给了新包裹。平时没事，一旦出事就是大事：你查物流，跳出两条记录；客服理赔，两个包裹共用一个号，谁是谁都说不清。

> 大白话
>
> 写软件的人都懂这个道理：空间要按「最坏的一天」来留，不能按「平常的一天」。双十一单日的处理量能冲到七亿件上下，是日常的一倍半。编号空间如果卡着日常量设计，那每年最忙的那天，恰好就是系统崩掉的那天。

所以多出来的位数，每一根都是一条伤疤：有的为了扛住双十一的洪峰，有的为了让单号隔得足够久才重复、旧件的记录早已归档，还有的纯粹是给还没发明的业务留座——万一以后要区分冷链件、国际件、同城件呢？号码长度一旦定死、印进几亿台扫码枪的固件里，想再加一位就是伤筋动骨。**编号空间的大小，是在设计那天就预付的保险费。**

这本书要回答的问题

一根快递单号，拆到最后其实是个哲学问题：世界为什么需要给一切编号？

答案藏在一个很日常的场景里。你对客服说「我的包裹」，客服看到的是三亿个在途包裹；你说「就是那个蓝色的箱子」，对方依然无从下手。直到你报出那十五位数字，三亿分之一的搜索瞬间变成了一次精确命中。**编号的本质，是把「那一个」从「那一堆」里无条件地挑出来**——不靠形容，不靠运气，不靠对方记性好。

包裹需要它，你的手机需要它（第九章会拆给你看），你银行卡里的钱能跨国旅行靠的也是它（第十章），连地球表面每一个点都有一对坐标当门牌（第十一章）。这本书就干一件事：把生活中这些不起眼的编号一个个拆开，看它们是怎么被设计出来的，又为什么是今天这副模样。

不过在出发之前，得先泼一盆冷水。你大概已经注意到，快递公司从来不用「张三寄给李四的蓝色箱子」来管你的件。名字为什么在大规模系统里必然失灵？纯粹唯一的编号又有什么副作用？这是下一章的事。

第二章 · 名字为什么不够用

第二章 · 名字为什么不够用

先做一个思想实验：把全国的快递、病人、包裹、订单上的编号全部抹掉，只允许用名字。世界还能转吗？

答案是不能，而且崩溃得比你想象得快得多。

一个班五十个人，就有两个张伟

2014 年有人用覆盖 11 亿人口的历史姓名数据做过统计：全国叫「张伟」的约有 **29.9 万人**，叫「王伟」的约 29 万，前十名每个名字都有二十多万人共用。注意，这不是抽样估算的误差，是实打实按户籍数出来的。

这意味着什么？假如医院只按名字挂号，北京一家三甲医院一天门诊上万人次，碰上两三个「张伟」是数学上的必然。护士喊一声「张伟，三号诊室」，走进去两个人，谁是来做胃镜的，谁是来拆线的？名字在村口管用，是因为村里只有三百人；人一过万，名字就从「标识」退化成了「线索」——它能帮你缩小范围，但永远锁定不了那一个人。

> 大白话
>
> 名字的作用是「在一群熟人里把你叫出来」，不是「在十四亿人里把你找出来」。这两件事听着像，其实是两种完全不同的工程问题。

歧义不止重名这一种

就算不重名，名字在大规模系统里还有一堆毛病：

- **同音不同字。**电话里报「我叫章伟」，对方可能写成「张卫」。口语传播天然带损耗。
- **名字会变。**改名、结婚改姓、起笔名、公司更名——名字是属性，不是身份。一个人的名字变了，人还是那个人。

- **写法不唯一。**同一个人，护照上是拼音 ZHANG WEI，系统里是「张伟」，老外登记成 Wei Zhang。三个写法，一个人。

所以大规模系统做的第一件事，就是把「指代」和「描述」分开：编号——一个人工分配的、只负责「唯一指向某个对象」的符号——接管指代，名字退回到描述。这就是编号的第一使命：**唯一性**，一个号码只对应一个对象，一个对象只对应一个号码。

你可能觉得这是废话——发个号谁不会？但现实里连国家都翻过车。中国的第一代身份证号码从上世纪八十年代开始发放，各地手工编制，结果到 2009 年公安部摸底时，全国竟有**171 万人**的身份证号码与别人重号。两个陌生人共用一个号码，意味着在银行、民航、社保系统里他们随时会被当成同一个人。公安用了近十年专项清理，到 2017 年才把重号人数从 171 万压到 **8 人**——剩下的 8 个人各有特殊困难改不动，最后只好在系统里给他们打上特殊标注，人工兜底。唯一性这条铁律，不是设计出来的，是付过学费才明白的。

那最简单的编号：1、2、3、4……

既然只要唯一，最省事的方案每个程序员都写过：自增 ID——数据库里来一个对象就发下一个整数，第 1 个是 1，第 2 个是 2，永不重复，永不枯竭（至少在 int 溢出之前）。

> 大白话
>
> 自增 ID 就像食堂排队发号：窗口只有一个，号一个一个往外发，谁也撞不上谁。前提是——窗口只有一个。

问题恰恰出在「窗口只有一个」上。自增 ID 的唯一性是靠**中央集权**换来的：必须有一个唯一的权威知道「上一个号发到几了」。小系统里这个权威就是数据库本身，没问题。可一旦业务长大，麻烦接踵而至：

- **一台机器发不过来。**每秒几十万订单进来，所有请求都排队等一个计数器，发号器成了全系统的咽喉。
- **分两台机器发，立刻撞号。**两台数据库各自从 1 数起，都发出一个「42 号订单」，合并数据时傻眼。想让它们不撞，就得互相通信协调——而协调本身，就是在重建那个慢吞吞的中央窗口。

- **号码会泄密。**你的订单号是 10086，竞争对手下单一看出 10090，就知道你今天只接了 4 单。连续编号把业务量拱手送人。

看出来了吗？「唯一」这件事，实现起来有两条路：**中心化发号**（一个窗口，简单但慢且脆弱）或者**让号码足够随机、足够长**，长到两台机器瞎编也撞不上（快，但号码变得没法读、没法记）。这就是所有编号系统要面对的第一对矛盾——唯一性想要集中，规模想要分散。

本章小结

名字是给人际关系用的，编号是给系统用的；系统规模一大，唯一性就是第一刚需。但「保证唯一」最便宜的办法—— 集中式地发连号——在规模面前要么成为瓶颈，要么泄露信息。于是工程师被逼向另一个极端：用超长随机数换取「不用协调也撞不上」的唯一。可那样的号码，人就再也读不动了。

唯一与可读，难道真的不可兼得？下一章，我们拿 UUID 和车牌号做一场对照实验，看看这对天生的冤家各自妥协在哪。

第三章 · 唯一与可读：一对天生的冤家

上一章结尾我们留下一个矛盾：编号的第一使命是唯一，可纯唯一的东西往往又臭又长，没人愿意看。这一章我们干脆做个对照实验：请出两位站在天平两端的极端选手。

一位从不重复的机器人

第一位选手叫 UUID（Universally Unique Identifier，通用唯一标识符），说白了，就是**一串随机生成的、长到几乎不可能撞号的字符串**。你在数据库、日志、手机 App 里随处能见到它，长这样：

`f47ac10b-58cc-4372-a567-0e02b2c3d479`

它一共 128 个二进制位，其中最常用的 4 号版本有 122 位是纯随机的（剩下 6 位固定用来标记「我是哪个版本」）。122 位随机意味着有多少种可能？2 的 122 次方，约等于 5.3 × 10 的 36 次方——把地球上每一粒沙子都数出来，再乘上一百亿亿倍，还没它大。

你可能会问：随机生成，不还是有撞号的概率吗？有，但小到离谱。这里有个著名的反直觉现象叫生日悖论：一个班只要 23 个人，就有五成概率有人同一天生日——因为两两配对的可能性增长得比你想的快得多。按同样的算法套到 UUID 上，算出 50% 撞号概率需要生成大约 2.7 × 10 的 18 次方个。换算成人话：

> 大白话
>
> 假设全球所有机器加起来每秒生成一百万个 UUID（这已经是相当高的负载），连着生成一百年，总量约 3 × 10 的 15 次方个。算式很简单：撞号概率约等于总量的平方除以两倍的总可能性数，即 $(3 \times 10^{15})^2 \div (2 \times 5.3 \times 10^{36}) \approx 9 \times 10^{-7}$，约百万分之一——跟一个人一年里被雷劈中的概率差不多。工程上，这就叫「不会发生」。但要注意，这个结论成立的前提是「总量控制得住」：真到了每秒生成十亿个的规模，一百年下来就会逼近 50% 那条线，所以 UUID 系统在高并发场景还得加重复检测兜底。

UUID 最妙的地方在于：**它不需要任何中央发号机构**。你电脑、我手机、某台服务器，各自随手一抓，就是全球唯一。没人登记，没人协调，没人批准。唯一性靠的是数学，不是管

理。

代价呢？显而易见：没有任何人能看一眼就记住它。它对人完全关闭，纯为机器服务。你跟客服说「我的订单是 f47ac10b-58cc……」，对方会在心里骂你。

一位生怕你看不清的选手

第二位选手在马路对面：中国车牌。

92式车牌是我国 1992 年定下来的车牌样式，结构是「汉字 + 字母 + 五位字符」，比如「沪A·D58K2」。拆开看：第一个汉字是省份简称（沪=上海），第二位字母是地市代码（A 一般留给省会或首府），后面五位才是这辆车自己的编号。

这个设计里每一条规则都在为「人」让路。举一个细节：车牌字符里**永远不会有字母 I 和 O**——因为 I 长得像 1，O 长得像 0，交警隔着五十米一眼扫过去，绝不能在「这是 1 还是 I」上犹豫。为了让人眼不误判，号段容量宁可砍掉两个字母。

再举一个：2016 年新能源车牌从 5 位升到 6 位，第一位固定用字母 D（纯电动）或 F（非纯电）。为什么要加这一位？因为老号段快发完了，而加一位之后容量扩大约 30 倍；为什么用 D 和 F 开头？因为这是拼音首字母，人一看就懂「这是电车」。

你看，车牌和 UUID 正好倒着来：

- UUID 把 122 位全部押给「绝不撞号」，人可读性归零；
- 车牌把号段容量一砍再砍（禁 I 禁 O、号段按城市分割），就为了让一个肉眼凡胎的交警在雨夜五十米外一眼认出。

分叉口：这个号码是给谁看的

这两个案例放在一起，编号设计的第一道岔路口就露出来了：**这个编号的主要读者，是机器还是人？**

注意我说的是「主要读者」。几乎没有编号是纯给机器或纯给人的，区别只在权重。怎么判断？有几条很实用的评判标准：

- **会不会被人念出来、抄下来？**会，就必须短、必须避开易混字符、最好带语义。订单号、车牌、快递单号都是这类。

- **会不会在屏幕上被并排比对？**会，就得考虑前缀区分度——UUID 撞号不怕，但前四位碰巧一样的概率不低，程序员对日志时看的其实只有开头几位。
- **生成它的地方，能不能方便地找中央机构协调？**能，就可以像车牌那样集中发号；不能（比如几十万个服务各自写日志），就只能学 UUID 用数学买唯一性。
- **号码里藏的信息，读者用得上吗？**车牌里的「沪A」对交警有用（一眼知道属地），UUID 里什么信息都没有——不是设计者懒，而是它的读者（数据库索引）根本不需要。

用一句话总结这对冤家：**唯一性可以向数学购买，可读性只能向设计购买；钱花在哪边，取决于谁在读。**

不过你可能已经注意到一个破绽：车牌说它可读，可它并不「绝对唯一」——「沪A·D58K2」只是在上海发牌范围内唯一，全国加起来靠的是「省份汉字」这一段前缀。换句话说，车牌没有追求 UUID 那种单点唯一，而是用「分层」的办法凑出了全局唯一：先看省，再看市，再看号码。

这个「先看大圈再看小圈」的思路，其实是编号系统能规模化的真正秘密武器。为什么一个号码装不下的时候，切几段就能装下了？电话区号和邮政编码会给我们答案——下一章见。

第四章 · 层级：把世界装进一棵树

第四章 · 层级：把世界装进一棵树

上一章结尾留了一个问题：一个号码要同时满足「唯一」和「可读」，位数很快就爆了。全球几十亿部电话，如果每部电话都直接从 00000000001 排到 99999999999，号码倒是够用，但没人记得住，也没人能维护——你想想，负责发号的那个人得记住几十亿个号码分给了谁。这个死局的解法，古人写信时就会了：没人把地址写成「中华人民共和国第 8742 万号住户」，而是写「江苏省南通市海门县……」——一级一级缩小范围，像走台阶一样走到目标跟前。这就是本章的主角：层级编号，把一个巨大的编号空间切成若干段，每一段只回答一个更小范围的问题。

电话区号：一台巨型分拣机的使用说明书

拨一个外地手机号，你拨的其实是 11 位数字。但打固定电话时，你得先拨区号。区号这个东西，就是一层明明白白的层级。

先看国际层。你给中国打电话要拨 `+86`，这个 86 不是随便抓的。上世纪六十年代，国际电信联盟（一个协调各国电信标准的联合国机构，你可以理解成「全球电话系统的总管委会」）把世界分成九个编号区：1 给北美，3 和 4 给欧洲，8 和 9 给亚洲……中国分到的是 8 区里的 86，香港 852、澳门 853、台湾 886，都是 8 字头一家人。第一层回答的问题是：**这通电话要去哪个国家？**

再看国内层，这里藏着一套几乎没人注意、但设计得相当讲究的体系。中国的固定电话网早年是四级汇接结构：最顶上是北京，区号 `010`——全国独一份的两位区号，自带首都特权；往下是八个大区交换中心，瓜分 `02X` 这一段：广州 020、上海 021、天津 022、重庆 023、沈阳 024、南京 025、武汉 027、成都 028、西安 029。再往下是普通地级市的三位区号，而且三位的数字不是乱排的：东北大区的城市，区号第二位跟着沈阳走 4 字头，比如大连 0411；华东跟着南京走 5 字头，比如苏州 0512；中南跟着武汉走 7 字头，比如长沙 0731。

大白话

翻译成人话：看到一个区号 0571，你不需要查表就能推理——5 字头，华东，南京大区的地盘；杭州的。区号本身就是一张压缩过的地图。

还有个耐人寻味的细节：`026` 是空缺的。这个紧挨着南京 025 的号码几十年没发出去，官方说法是预留，民间最流行的猜测是给台北留的——因为台湾在行政区划序列里常被归入华东，025 的下一个就是 026。真假姑且不论，这件事本身就说明了一个道理：**层级编号一旦建立，空位也是有含义的**，发号的人得为还没出现的城市留位置。

美国人玩的是同一套思路的另一个版本。1947 年北美编号计划给全美分区号时，还在用转盘电话——手指插进孔里拨到底，数字越大，转盘回弹越慢，拨一个 0 要等将近一秒。于是设计者把「回弹最快」的号码给了人口最多的城市：纽约 212（总脉冲数只有 5），洛杉矶 213，芝加哥 312。而人烟稀少的地区领到了 907（阿拉斯加）这种慢吞吞的号码。今天转盘早进博物馆了，但曼哈顿的老号码 212 至今是身份象征——**一段编号的设计逻辑，会凝固在号码里，比设计者活得还久**。

邮政编码：六位数里的四级台阶

区号是给交换机看的层级，邮编是给分拣员看的层级。中国的邮政编码是六位，实行「四级六码制」，每一位都有明确分工：

- 前两位：省。比如 `22` 是江苏，`13` 是吉林。
- 第三位：邮区，一个省内按邮路划的大片。
- 第四位：县或市。
- 最后两位：投递局，也就是最终把信送到你手上的那个邮电所。

拿真实的例子拆一遍：`226156` ——22 江苏，6 南通邮区，1 海门县邮局，56 具体投递局。一封信从外省进江苏，分拣设备只读前两位就能决定它上哪趟火车；到了南通，再看第三位、第四位往下分。每一级分拣员只需要认识编号的一段，不需要认识全国几十万个地名。

大白话

翻译成人话：六位邮编不是「一个号码」，而是「四个问题排队等着回答」——哪个省？哪个邮区？哪个县？哪个投递局？每回答一个，剩下的世界就小一圈。

层级真正的杀手锏：把发号的权力下放

到这儿你可能会说：层级不就是把一个大号码拆成几段吗，有什么了不起？了不起的地方不在号码本身，而在**组织方式**。一个扁平编号系统必须有一个「全知全能」的中央发号员，所有号码都从他手里过；而层级系统里，国际电信联盟只管把 86 发给中国，中国邮电部门只管把 021 发给上海，上海市内的号码怎么分，上海自己说了算。**每一级只需要对自己那一段保持唯一，谁也不打扰谁。**

软件工程师对这套东西熟得很：文件路径 `/home/wjs/book/04.html` 就是层级编号，操作系统维护根目录，你维护你自己的 `home`。快递公司内部分单号也是这么干的——总部给每个转运中心划一段号段，转运中心再往下放给网点，谁也不用等总部审批。层级让编号系统从「一个中央机构的记账本」变成「一棵可以无限挂节点的树」，这是它能规模化的根本原因。

但树也有树的烦恼。电话区号绑死的是地理，可万一编号想表达的不是「在哪儿」呢？比如一台服务器的地址，它属于哪个「网络」，跟它在哪个城市是两回事。下一章我们会看到，互联网干脆给同一台机器上了两套户口——一套按物理拓扑分层给路由器用，一套按管理权分层给人用——而这两套层级之间，还需要一个全天候的翻译官。

第五章 · 域名与 IP：一台机器的两套户口

第五章 · 域名与 IP：一台机器的两套户口

上一章我们说到，层级的妙处在于把分配权一级一级下放。这一章我们看这套思想最漂亮的一次实战：互联网上同一台服务器，同时揣着两套编号——`142.250.72.14` 和 `google.com`。它不是历史遗留的冗余，而是两套刻意分开的层级，各自回答完全不同的问题。

IP 地址：给路由器看的层级

IP 地址，说白了，就是互联网上每台设备的门牌号，32 个二进制位，通常写成四段十进制数字，比如 `142.250.72.14`。约 43 亿个，一个萝卜一个坑。

关键在于：IP 的分段不是按「谁拥有这台机器」分的，而是按**网络拓扑**分的——也就是这台机器物理上接在网的哪个位置。前面几段大致圈出「哪个大洲的哪个运营商」，中间圈出「哪个城市的哪个机房」，最后几段才是「这个机房里的哪台机器」。

为什么这么分？因为 IP 的主要读者不是人，是路由器。路由器就是网络世界里的分拣员：每收到一个数据包，它只看地址开头几段，就知道该往哪条干线扔，根本不用关心全世界 43 亿台机器各自在哪。

> 大白话
>
> 这就像邮局分拣信件：上海的分拣员看到「四川省成都市……」，只负责把信扔进「成都」的邮袋，至于玉林小区 3 栋 2 单元在哪，那是成都分拣员的事。每一级只需要记住下一级怎么走——这就是层级给路由器省下的亿级规模的记忆负担。

所以 IP 地址本质上是**一张按地理位置和线路结构编出来的地图坐标**。机器搬个家、换条线路，地址就得换。它对机器友好，对人刻薄——没人记得住一串点分数字。

域名：给人看的层级

域名则是另一套逻辑。拿 `mail.google.com` 举例，注意它要**从右往左读**：`com` 是顶级域，回答「这归哪个登记机构管」；`google` 回答「这家公司叫什么」；`mail` 回答「这家公司内部的哪项服务」。每一段代表一层**管理权**，而不是一层地理位置。

看出区别了吗？IP 按「网线怎么连」分层，域名按「谁说了算」分层。Google 可以把服务器放在俄勒冈、新加坡或芬兰，IP 地址随便换，域名纹丝不动——因为域名登记的是「归谁管」，而这件事不会因为搬家改变。

两套户口，各管各的：一套给机器的路由用，一套给人类的记忆用。中间总得有个翻译。

DNS：互联网的电话簿

DNS（域名系统）就是这个翻译官：你输入 `google.com`，它负责查出对应的 IP 地址，然后把查询结果缓存起来，下次更快。

它本身也是一个层级系统，而且是人类历史上规模最大的**分布式数据库**——没有一台中央服务器存得下全球 3.6 亿多个已注册域名（Verisign 2024 年统计约 3.62 亿，且还在增长），所以查询也是一级一级问的：

1. 先问根服务器：全球逻辑上只有 13 组（用字母 A 到 M 命名），它们不存具体域名，只回答「`.com` 的事归谁管」；
2. 再问 `.com` 的顶级域服务器：「`google.com` 归谁管」；
3. 最后问 Google 自己的权威服务器：「`google.com` 的 IP 是多少」。

每一级只知道自己下一层的电话。没有谁能背下整本电话簿，但任意一级只要指对方向，答案就一定能被问到。**分层在这里不只是省记忆，更是省协调**：Google 改自己的 IP，只需要更新自己那一级的记录，根服务器和其他所有人完全无感。这正是上一章说的「分配权下放」的终极形态——连「查询权」也下放了。

为什么 IP 会用完，域名不会

现在回答本章标题引出的那个问题。**编号空间的大小，也是设计出来的。**

IPv4 只有 32 位，上限约 43 亿，写死在协议里了。1980 年代设计它的时候，没人相信世界上会有 43 亿台联网设备——那时整个 ARPANET 只有几百台主机。结果是：2011 年 2 月 3 日，负责全球发号的 IANA 把最后五个地址块分给了五大洲的区域机构，宣布总池子见底；两个多月后，4 月 15 日，亚太地区的 APNIC 第一个把自己那份也发光了。现在一个新机房想拿一段干净的 IPv4，只能去二手市场买，一个地址被炒到几十美元。

域名为什么永远不会用完？因为域名是**变长的**。层级里每一段都可以随便加长：好名字被占了，你总可以起个更长的——`great-name-2024-shop.com` 永远有空位。而 IP 的每一段是固定 8 位，一格都不能多。**定长换效率，变长换容量，这是编号设计里一对永恒的取舍。**路由器为了每微秒转发一个包，必须让地址定长好查表；人类为了永远有新名字可用，宁可让域名长短不一。

互联网的补救方案你也猜到了：IPv6，把地址从 32 位一口气扩到 128 位，数量约 3.4×10^{38}——多到可以给地球上每粒沙子都发一个号。但那是另一个「兼容性」的故事了，我们第十章讲跨国银行编号时会再碰到这类「旧编号发出去就收不回」的麻烦。

两套户口的启示

回头看，域名和 IP 这对组合回答了一个所有编号系统都会遇到的问题：**当「机器要的规律」和「人要的规律」不是同一种规律时，别硬凑——分两套编号，再造一个翻译层。**

下一章我们换个方向：有些编号不止标识身份，还把规律直接写进了号码里——为什么 CZ 奇数航班号往南飞，为什么美国个位数带 5 的州际公路都是大动脉？会说话的编号，不用查表就能推理。

第六章 · 会说话的编号：航班号与高速公路

1944 年 7 月的一个傍晚，如果你是驻守上海虹桥机场的调度员，听到无线电里报出「国航 1501，请求进近」，你其实什么也不知道——1944 年还没有航班号。但如果你身处今天的塔台，听到 `CA1501` 这个编号，你不查任何表，就能推出一串事实：这是国航的飞机，从北京基地出发向外飞，多半是个重要航线。这就是本章的主角：**会说话的编号**。

前几章里，编号要么追求唯一（UUID），要么追求层级（域名、IP）。但还有第三条路：**把规律直接编码进号码本身**，让内行看一眼就能推理，不用查表。航空和公路系统把这种思路用到了极致。

航班号：六位字符里藏着一张时刻表

拆开 `CA1501`。前两位 `CA` 是航空公司代码，由国际航空运输协会统一分配——中国国际航空是 CA，东航是 MU，南航是 CZ。这不难理解，本质上是第四章讲过的「层级下放」：协会只发两位字母，后面的数字归各公司自己编。

妙处全在后面的数字里。中国民航有个沿用多年的惯例：**单数是去程，双数是回程**。所谓去程，就是「从基地向外飞」。比如 MU508 是从香港飞回上海——东航的基地在上海，这是回家，所以用双数。一架飞机早上从基地出去（奇数号），晚上回来（偶数号），两个号码天然成对，调度员看到奇偶就知道这架飞机现在是「在外」还是「回家」。

数字大小也有讲究。传统上，位数越少、号码越靠前，航线越重要——国航的 1 字头多是旗舰国际线。这不只是虚荣心：短的号码在无线电里读起来快，在人工排班表上排得靠前，是一套「重要的事优先」的朴素设计。当然，这些只是惯例而非铁律，例外不少——你偶尔也能看到挂着奇数号往回飞的航班。编号会说话，但它说的是方言，不是法律。

大白话

翻译成人话：航班号像部队番号。番号里带着建制——一听「三连二排」，老兵就知道这是哪支部队、大概什么任务；外行听起来只是一串数字。

中国高速：G 字头背后的一整张地图

天上的编号说给调度员听，地上的编号要说给司机听——而且要在时速 120 公里、只有两三秒瞥一眼路牌的情况下说清楚。中国国家高速公路的编号，是这种极限场景下的杰作。

规则只有三条，背下来你就是内行：

- **G 加一位数，是首都放射线。**G1 京哈、G2 京沪、G3 京台……一直到 G7 京新（乌鲁木齐）。以北京为圆心，从东北方向开始按顺时针排号——G1 指向东北，G7 指向西北，你在哪条放射线上，看号就知道自己大致在北京的哪个方位。
- **G 加两位奇数，是南北纵线；两位偶数，是东西横线。**G15 沈海高速（沈阳—海口）纵贯中国东部海岸线，是奇数；G42 沪蓉高速（上海—成都）横贯东西，是偶数。纵线从东向西编号递增，横线从北向南递增。
- **S 打头是省级高速**，编号规则各省自定——这是层级下放又一次出现。

这套 2009 年推行的全国统一编号，改掉的是一笔老账：以前一条路跨省就改名，京石高速、石安高速、安新高速……一条北京到深圳的动脉能被切成十几个名字，司机完全无法建立「这条路通往哪里」的整体直觉。改名之后，G4 京港澳高速一个号码从北京直通香港口岸，**编号把碎掉的路重新拼成了一张网**。

美国州际公路：把规律写到个位数上

更早做这件事的是美国人。1956 年艾森豪威尔签署法案开建州际公路系统（Interstate System）时，顺手设计了一套至今堪称范本的编号：

- 奇数是南北向，从西向东递增——所以 I-5 贴着西海岸（圣地亚哥一路向北到加拿大），I-95 贴着东海岸（迈阿密到缅因）。
- 偶数是东西向，从南向北递增——I-10 在最南边（佛罗里达到加州），I-90 在最北边（西雅图到波士顿）。
- **个位数是 0 或 5 的，是横贯大陆的大动脉。**I-5、I-95、I-10、I-80、I-90，全是国家级干线。

还有更精巧的一层：三位数的州际公路是「支线」，后两位标明它侍奉哪条主线。I-405 是 I-5 的支线（洛杉矶人听到「the 405」会露出痛苦的表情）。第一位数字还要区分功能：奇数

开头表示扎进市区的支线，偶数开头表示绕城或环城路。一个老司机看到 `I-280`，不用地图就知道：这是 I-80 的一条环城线。

> 大白话
>
> 翻译成人话：这套编号像地球仪上的经纬网格被烙在了公路上。奇偶告诉你方向，大小告诉你位置，0 和 5 告诉你分量。号码本身就是一张缩略地图。

编号即文档

把三章串起来看：第二章说编号首先要唯一，第四章说放不下就分层级，本章说**层级里的每一格都可以再塞进语义**。会说话的编号有一个共同的好处：它把「使用说明书」压缩进了号码本身，系统的老用户——调度员、卡车司机、程序员——凭记忆就能推理，而不必每次查询权威数据库。这在大规模、高并发、人命关天的场景里，省下的不只是时间，还有出错的机会。

但它也有代价：语义是约定，约定会过时。航线会调整，城市会扩张，当年「从东向西递增」的整洁网格，几十年后总会被新修的支线戳出窟窿。好的编号系统必须为「未来的例外」留出余地——这个主题我们到第十章讲跨国银行编号时再算总账。

在此之前还有一个更基本的问题没解决：航班号会说话了，可人还是会抄错。值机柜台把 CA1501 听成 CA1507，一个字之差，行李就飞去了另一座城市。编号光是聪明还不够，它还得会**自查**——这就是下一章校验位的故事。

第七章 · 校验位：编号自带的防伪术

第七章 · 校验位：编号自带的防伪术

先做个思想实验。银行柜员输入你的 16 位卡号，手一滑，把其中一个 3 敲成了 8。如果这个号码是纯粹的流水号，系统会高高兴兴地收下它——钱是实打实扣了，只是记在了别人的账上。可现实中，这种转账几乎不会发生，因为卡号的**最后一位根本不是卡号的一部分，它是前 15 位算出来的一个「指纹」**。改任何一位，指纹就对不上，系统在钱动之前就把这笔单子拦下了。

这个指纹有个正式名字：校验位，就是编号末尾专门留出来的一位（或几位），它不携带任何业务信息，唯一的作用是根据前面所有位重新算一遍，验证这串数字在抄写、录入、传输的路上没有出错。

亲手算一遍你的身份证第 18 位

光说原理不过瘾，我们拿国家标准 GB 11643 里的示例号码 `11010519491231002X` 真算一遍。前 17 位每一位都有一个固定的「权重」，从左到右依次是 7、9、10、5、8、4、2、1、6、3、7、9、10、5、8、4、2。算法只有三步：

1. 每一位乘上自己的权重再相加：1×7 + 1×9 + 0×10 + … + 2×2 = **167**。
2. 拿 167 除以 11，余数是 **2**。
3. 查一张小表：余数 0→1，1→0，2→X，3→9……一直排到 10→2。余数是 2，所以第 18 位是 **X**。

对了，这个号码的尾号确实是 X。整个过程小学生都能算，但设计里藏着两个机关。

第一个机关：为什么除以 11，而不是好算的 10？因为 11 是质数。除以 10 的话，校验位只看总和的个位，权重稍不注意就会「撞车」；而质数 11 和每个权重都互质，保证了**任何一位数字变动、任何相邻两位交换，余数几乎必然改变**。

第二个机关：为什么有人身份证尾号是 X？余数有 11 种可能（0 到 10），但一位十进制数字只有 10 个，装不下。于是标准借用了罗马数字的 X 来表示 10——你的尾号 X 不是什么

特殊身份，只是余数恰好落在了那个装不下的格子里。

大白话

说人话：校验位就像你出差报销时，财务让你把发票金额逐笔报一遍，再报一个合计。他不用重算每一笔，只要核一下合计对不对——对不上，中间一定有哪笔抄错了。权重和质数除法的作用，是让「抄错一笔」和「两笔写反」这两种最常见的错，都一定会把合计弄乱。

信用卡的 Luhn 算法：一个 1954 年的小发明

身份证那套加权求和叫 ISO 7064 MOD 11-2，而你的银行卡、信用卡用的是另一套更著名的算法：Luhn 算法，IBM 工程师 Hans Peter Luhn 在 1954 年申请专利的小发明。它更简单：从右往左，每隔一位把数字翻倍（翻倍后超过 9 就减 9，比如 8×2=16→7），全部加起来，总和能被 10 整除就算合法。

比如验证 `7992739871`：翻倍位处理后再求和得 67，距离下一个 10 的倍数还差 3，所以这张卡的完整卡号末位必须是 3，即 `7992739873`。你现在随便打开一个支付页面，乱敲 16 位数字，页面不用联网就能立刻告诉你「卡号有误」——靠的就是这个本地一秒能算完的校验。

Luhn 的战绩很能说明设计目标：**它能抓住 100% 的单数字错误，以及几乎全部相邻两位写反的错误**——唯一的漏网之鱼是把 09 写成 90（或者反过来），因为这一对交换后总和恰好不变。这不是疏漏，而是取舍：人类录入最常见的错误就是「打错一位」和「相邻两位颠倒」，这两类占了录入错误的绝大多数。用一个谁都能心算的算法拦住它们，性价比极高。想抓得更全可以用更复杂的算法（比如 Verhoeff 算法能抓住所有转置错误），但代价是验算变重——而卡号每秒要被全世界的终端验证几十亿次，轻，就是正义。

它防得了手滑，防不了伪造

这里必须划清一条边界，也是本章最想让你记住的一句话：**校验位是「检错」，不是「防伪」**。算法是公开的，任何人都能给自己的假号码算出一个完全合法的校验位。校验位能回答的问题是「这串数字在传输过程中有没有被弄错」，而不是「这串数字是不是权威机构真发过的」。

所以书号 ISBN 的最后一位（ISBN-10 时代同样可能出现 X）、快递单号的末位、银行卡的末位，它们防的是同一个敌人——**无心之失**。对付有心之人，得靠另一套体系：登记机构的数据库里查不查得到这个号。那是后面章节的事。

不过校验位思路再妙，也只是「抄错了能发现」。工程师的下一个想法很直接：人抄写本来就不靠谱，**能不能干脆不让人抄，让机器自己去读？**超市收银台「嘀」的那一声，就是答案——下一章，我们去看条形码和二维码如何把编号从「给人看的数字」变成「给光看的图案」，以及二维码为什么缺了一角还能扫出来。

第八章 · 条形码与二维码：让机器替你读

第八章 · 条形码与二维码：让机器替你读

上一章的校验位是个很朴素的方案：既然人一定会抄错号码，那就多写一位，让机器当场算一遍。但你有没有想过——为什么非得让人抄？收银员敲键盘、库管员填单子，这两个动作才是错误的源头。校验位是亡羊补牢，而条形码的思路更狠：**把号码变成一幅图，让人从头到尾别碰它。**

这就是条形码（barcode）的出发点——用一排宽窄不一的黑条白缝，把一串数字编码成光线能直接"看"到的东西。它不在编号上做任何加法，它改变的是编号的载体：从给人看的符号，变成给光看的图案。

条纹为什么这么画

随便拿起手边一件商品看它的条形码，你会发现两件事：条纹只有黑和白，没有灰；条纹有的粗有的细，但粗细就那么几种。这不是审美选择，这是为一件非常具体的事情设计的——激光扫过去的那一瞬间。

收银台的扫描枪发出一道红光横着扫过条形码，黑条吸光，白缝反光，接收器看到的其实是一串"暗—亮—暗—亮"的电压起伏。机器根本不"看"条纹，它只听这串起伏的节奏：暗持续了多久，亮持续了多久。**每个数字占 7 格等宽的模块，由"黑—白—黑—白"四段拼成**，每段宽窄不一，窄的只占 1 格，最宽的占 4 格。不同数字就对应不同的宽窄排列——机器量出这四段各占几格，一查表就知道是哪个数字。换句话说，条形码不是一种图案，它是一种摩尔斯电码，只不过点划换成了宽窄。

为什么不用灰度？因为印刷会糊。1972 年 RCA 和克罗格超市在辛辛那提一家门店做过一次著名的真实试点，把条码印成同心圆靶（绰号"牛眼码"），想法很妙——圆形从哪个方向扫都能读。结果不尽如人意：印刷机稍微洇一点墨，圆环就容易糊成一片，分不出环的边界。牛眼码最终没能成为标准——1973 年，超市行业专门成立的委员会在各家提交的方案里反复比较，最后选定了 IBM 设计的线性条纹码，也就是今天的 UPC。这个决策考虑的因素很

多，印刷容错是关键的一条：条纹严格平行，即使墨洇开了，黑条只会变粗变细一点点，而黑白之间的边界依然锋利。好设计往往先想清楚"最坏情况下它怎么坏"。

条形码里还藏着一处从前面几章借来的设计：开头、中间、结尾各有一组略长的条纹，叫分隔条——相当于给机器的路标，告诉它"从这里开始读，读到中间别停，到这里结束"。有了路标，扫描枪不管从左往右还是从右往左扫，都能把号码正过来。层级、分隔、校验，这些机制在这本书里反复出现，因为它们是解决同一类问题的同一类答案。

一包口香糖的清晨

1974 年 6 月 26 日早上 8 点 01 分，俄亥俄州特洛伊市一家 Marsh 超市，收银员把一包绿箭口香糖（Wrigley's Juicy Fruit）放到了扫描台上——这是人类历史上第一件被条形码卖出的商品。那包口香糖如今收藏在史密森尼博物馆。

值得记住的不是这个仪式感时刻，而是它之前的三十年。条形码的专利 1952 年就被伍德兰和西尔弗拿下了——据说灵感来自他在沙滩上把摩尔斯电码的点划拉长成了条纹。但之后的二十年几乎没有商店愿意用，因为条形码是一个典型的**鸡生蛋系统**：没有扫描枪，厂家不愿印码；没有商品印码，商店不愿买扫描枪。最后打破僵局的是行业协会定了全美国统一的标准（UPC），所有厂商印同一套码，所有扫描枪读同一套码——编号系统最值钱的从来不是技术，而是"所有人都认它"这个共识本身。

二维码：把电码铺成围棋盘

条形码能装下大约 20 个字符，管一盒口香糖绰绰有余，但到了 1990 年代的丰田工厂就不够用了。当时的汽车零部件要追踪零件号、批次、产地，工人得在一个箱子上贴好几张条码、扫好几次。丰田子公司 Denso Wave 的工程师原昌宏（Masahiro Hara）被派去解决这个烦人事，两年后，1994 年，他交出了答案：二维码（QR Code，QR 是 Quick Response，"快速响应"的缩写）——把条形码那排一维的条纹，铺成二维的方格。

据说原昌宏的灵感来自午休时下的围棋。围棋盘的黑白格点天然就是一种二维编码：条形码只能在一个方向上排黑白，方格在两个方向上都能排，同样的面积装的信息直接翻两个数量级——标准二维码最多能装 7089 个数字，写下一整段网址还有富余。

你注意过二维码三个角上那三个"回"字形的大方块吗？那叫定位符——手机摄像头靠这三个锚点瞬间判断码在哪里、转了多大角度、歪了多少，然后把它"掰正"成一个标准的方格网，

再逐格读出黑白。三个锚点在三个角而不是四个，是故意的：第四个角留空，机器一眼就知道哪边是"上"，你把码转 90 度它也能认出来。

缺了一角为什么还能扫出来

二维码最神奇的本事是：贴纸被撕掉一块、被咖啡渍盖住一角，甚至中间被人印了个 logo，它照样能扫出来。这不是运气，是它出厂时就自带了"备份"。

二维码内部用一种叫里德-所罗门纠错（Reed-Solomon error correction）的数学方法存数据——大白话说，它把要存的信息打碎成许多小块，再额外算出一批"冗余块"一起存进去；读的时候，就算丢了一部分块，靠剩下的块也能反推出丢失的内容。就像拼图少了几片，但每片边缘都有和邻片重复的画面，缺的画面能推回来。

更妙的是，冗余的量是**可调的设计参数**，不是固定值。二维码有四档纠错等级：L 档允许约 7% 的面积损坏，M 档 15%，Q 档 25%，H 档 30%。印在小卡片上的码用低档，省地方；印在户外广告牌上、天天风吹日晒的码用高档，宁可多花格子也要买容错。你在餐厅扫码点单的那个码如果中间有店家的 logo，它几乎一定是 H 档——设计师早就把"要被 logo 盖住"这件事算进了冗余里。第七章的校验位只能告诉你"这串数错了"，而纠错码能告诉你"错了，而且正确答案是这个"——这是检测和修复的区别，一个报警，一个自愈。

> 大白话
>
> 一句话总结这两代编码的关系：条形码说"别让人抄"，二维码说"别让图坏"。前者消灭录入这个环节，后者假设录入之后还会遭遇磨损、污渍和破坏，并提前买好保险。编号走到这一步，已经不只是"唯一地标识一个东西"，而是"在糟糕的现实世界里可靠地存活下来"。

Denso Wave 还做了一个关键决定：手握二维码的专利，但宣布不收专利费、开放给所有人用。三十年后，这个原本为汽车零件设计的小方格，成了中国街头从煎饼摊到公交站都在用的基础设施——又一次，编号的价值来自共识，而开放是获得共识最快的方式。

不过你有没有发现一个问题：到现在为止，我们给商品编号、给信息编号，可这些号码都印在包装和屏幕上。东西本身呢？你手里这部手机出厂时就被烙上一个全球唯一的"身份证号"，不归任何商店管，却能让全世界的网络认出它。谁有权给全世界每一台设备发号？下一章，我们去看看编号背后的"户籍科"。

第九章 · MAC 与 IMEI：给硬件上户口

第九章 · MAC 与 IMEI：给硬件上户口

前面几章编号的对象都是「信息」——一个域名、一个航班、一本书。这一章我们给「东西」编号：一块网卡、一部手机、一台出厂的笔记本。给信息编号，错了大不了改数据库；给东西编号，号码是烧进硬件、跟着产品流向全世界的，收不回来。所以这类编号系统背后都有一个绕不开的问题：**谁有权发号？**

MAC 地址：出厂时的胎记

你的电脑插上网线、连上 Wi-Fi，网卡开口说的第一句话里就带着自己的 MAC 地址——一串 48 位的二进制数，通常写成 12 个十六进制字符，比如 `3C:22:FB:8A:4E:01`。它是网卡出厂时烧进去的「胎记」，作用是在同一个局域网里区分「这块网卡」和「那块网卡」。

拆开这 48 位，你会看到一个熟悉的结构——层级。前 24 位叫 OUI（Organizationally Unique Identifier，组织唯一标识符），说白了就是「厂商代码」，由 IEEE（电气电子工程师学会）下属的注册机构统一卖给厂商。后 24 位由厂商自己往下发，爱怎么编怎么编，只要别重复。IEEE 是「公安局户政科」，厂商是「街道派出所」，一块网卡的号码是两级登记的结果。

这套体制很有意思的地方在于：**IEEE 发的不是号码，而是发号的权力。**厂商花钱买一个 OUI，买到手的是 24 位后缀空间——也就是 1677 万个号码的分配权。苹果买了很多个 OUI，所以你拿任何一台苹果设备的 MAC 前三节去查公开的注册表，都能查到「Apple, Inc.」以及它在库比蒂诺的注册地址。编号本身就是一张公开的户口本。

大白话

为什么非要有个「户政科」？因为 48 位里前 24 位如果不集中登记，两家厂商就可能撞车：你烧进去的「3C:22:FB」开头，地球另一端另一家公司也在用。同一家公司内部自己保证不重复很容易，跨公司不重复只能靠一个所有人都认账的中央登记处。权威不是从数学里来的，是从「大家都认它」来的。

厂商代码太大手大脚地用，IEEE 后来也学会了「按流量收费」：大户买 MA-L（1600 万个号），中等需求买 MA-M（100 万个），小玩家买 MA-S（4096 个）。号码空间是有限的资产，登记机构就是资产管理人。

一场由「胎记」引发的隐私战争

MAC 地址的设计假设是：号码固定不变，全球唯一。这在工程上是优点，落到人身上就成了麻烦。2014 年前后，有零售商被发现干这么一件事：在商场里布 Wi-Fi 探针，被动接收路过手机不停广播的 MAC 地址——你进店、在哪个货架前站了多久、隔几天又来一次，全被一个固定不变的号码串成了轨迹。手机没连上任何 Wi-Fi，只是开着，就已经在「报到」。

苹果的回应在 2014 年的 iOS 8：手机在扫描周围 Wi-Fi 时不再广播真实 MAC，而是每次随机生成一个假 MAC。安卓和 Windows 随后跟进。这等于承认了原设计的失败：**「全球唯一且永久不变」对一个硬件标识符来说太过分了**，于是系统主动把唯一性打了个折——只在通信建立之后才亮出真身（现在连建立之后也常常用针对每个网络单独生成的随机地址）。

这给编号设计添了一条前面没出现过的原则：一个号码被滥用的方式，设计师在画图那天根本想不到。所以好的编号系统要预留「翻脸」的余地。

IMEI：15 位数字管起全球几十亿台手机

MAC 管的是「网卡」，可一台手机对移动网络来说还有另一个身份：整机。在手机拨号盘输入 `*#06#`，屏幕会跳出 15 位数字——IMEI（International Mobile Equipment Identity，国际移动设备识别码），即这台手机的全球唯一编号，跟插哪张 SIM 卡无关。SIM 卡标识的是「你这个人/这份合约」，IMEI 标识的是「这台铁」。

15 位数字分三段，每段背后是一个角色：

- 前 8 位是 TAC（Type Allocation Code，型号分配码），由 GSMA——全球移动运营商的行业协会——在设备送审认证时分配，精确到品牌和具体型号。网上随便一个查询工具，输入前 8 位就能告诉你这是哪款手机。
- 中间 6 位是流水号，厂商自己在这个型号内部往下数。
- 最后 1 位是校验位，用的就是第七章算过的 Luhn 算法——同一个数学小机关，又在这里站岗。

注意这里的发号链条比 MAC 多了一环：**GSMA 不直接面对厂商，它把 TAC 的分配委托给各地的认证机构**，厂商过认证时顺带拿到号码段。层级又深了一层，但逻辑和第四章的电话区号一模一样：每一段回答一个更小范围的问题，分配权逐层下放。

IMEI 最硬的一次实战是治偷手机。早期的对策是停机、停 SIM 卡——但小偷换张卡就活了。后来的做法是全球运营商共享一份黑名单：手机一上报失窃，这台设备的 IMEI 进库，全世界的运营商拒绝为它提供网络服务。换十张 SIM 卡也没用，因为网络认的是设备号码，不是卡。一台「砖头」不好销赃，偷盗的动机就被釜底抽薪了。这个机制能成立，恰恰因为 IMEI 绑定的是硬件本身——这就是给「东西」编号的独特威力。

编号系统的第三块拼图

把这章和前面串起来看，一套编号系统要回答三个问题：**号码怎么设计**（唯一性、层级、校验，前八章的事）、**给谁编号**（信息还是实物），以及本章的主角——**谁来发号**。MAC 的户政科是 IEEE，IMEI 的户政科是 GSMA，序列号的户政科是厂商自己。**每个运转良好的编号系统背后都站着一个登记机构，它的权威来自全行业的承认，而不是来自技术本身。**技术可以设计，承认只能经营。

可问题来了：IEEE 和 GSMA 之所以能发号，是因为整个行业愿意坐下来听一个机构的。如果面对的不是一个行业，而是几十个国家、几十套早已各自运转的编号系统——比如各国长得完全不同的银行账号——没有谁能当「全球户政科」，这些系统之间怎么互相转账、互相认账？下一章，我们看编号世界里最艰难的一场握手。

第十章 · SWIFT 与 IBAN：编号如何跨国握手

第十章 · SWIFT 与 IBAN：编号如何跨国握手

前面九章，我们给自己熟悉的东西编号：快递、书、电话、银行卡、域名。这些系统大多是「一个人说了算」——邮政总局定邮编，ISO 定 ISBN，IEEE 定 MAC 前缀。一家权威，一套规则，从头设计，干净利落。

现在来到全书最难的场景：你想给「全世界的银行账户」编号。问题是，每个国家早就有了自己的一套——中国的卡号 19 位，美国的账号加路由号，英国的账户配 6 位排序码，德国的账号位数各家银行都不一样。这些系统已经跑了几十年，底下躺着几十亿个真实账户。你**不能推倒重来**，只能想办法让这些编号「互相转账」。

这一章讲的就是工程世界里最贵的一条经验：兼容旧系统，比设计新系统难十倍。

先解决「找银行」：SWIFT 代码

跨国转账的第一步不是转钱，是回答一个问题：钱要寄给**哪家银行**？全世界有上万家银行，「中国银行」和「美国银行」名字像，「花旗」在纽约和伦敦的又是同一家——光靠名字肯定出事（第二章说过的道理）。

答案是 BIC（Bank Identifier Code，银行识别码），因为由一个行业组织维护，大家更习惯叫它 SWIFT 代码——SWIFT 是「环球银行金融电信协会」的缩写，1973 年由 15 个国家的 239 家银行凑起来成立的合作社，专门给银行之间传报文。一个 SWIFT 代码长 8 或 11 位，比如中国银行北京总行的 `BKCHCNBJ`：前 4 位是银行（BKCH = Bank of China），第 5、6 位是国家（CN），第 7、8 位是城市或地区（BJ），后面可选的 3 位是具体分行。还是第四章那套层级思想：每一段回答一个更小范围的问题。

这里有一个流传极广的误会必须拆开：**SWIFT 一分钱都不碰**。它不是「钱的管道」，而是一家电报局——它手里没有任何资金，只是把「请从 A 账户付 5 万美元给 B 账户」这样的指令安全地送到对方银行，真正的记账发生在两家银行之间。SWIFT 干的是 Telex（电传打字机）当年的活儿，只是更快、更标准、更难伪造。所以 SWIFT 代码的本质是**银行的收件地**

址，不是钱本身。知道这个区别，你再看新闻里「把某国踢出 SWIFT」就明白了：那不是冻结资产，是拔掉银行的电话线——钱还在，但银行收不到指令，生意做不成了。

再解决「找账户」：IBAN 的套娃智慧

银行找到了，还得找到具体账户。麻烦在于各国账号格式完全不同：位数不同，有的带字母，校验方式不同，而且账号里哪些位是分行代码，各国规则也不一样。九十年代跨境转账的错误率高得惊人，错一笔就要人工处理、退钱、扯皮。

1997 年，ISO 发布 ISO 13616 标准，推出 IBAN（International Bank Account Number，国际银行账户号码）。IBAN 的设计思路用两个字概括：**包住**。它不废除任何国家的旧账号，而是在外面套一层国际通用的「信封」：

```
GB29 NWBK 6016 1331 9268 19
```

这是一个英国账户的标准示例。拆开看：前两位 `GB` 是国家代码；接下来两位 `29` 是校验位（第七章讲过的数学小机关，IBAN 用的是 mod 97，整除余 1 才算合法，能拦住绝大多数抄错）；剩下的部分 `NWBK 6016 1331 9268 19` 叫 BBAN——基本银行账户号，**里面装的就是英国原来的账号**：NWBK 是 NatWest 银行，601613 是分行的排序码，31926819 是账号，原汁原味，一位没动。

> 大白话
>
> 翻译成人话：IBAN 就像国际快递的运单袋。你原本国内寄件的面单不用换，国际段只是在外面套一个透明袋，袋子上用全世界都认的格式写上「发往哪个国家、防伪码是多少」。到了英国口岸，英国人撕开袋子，里面还是他们看得懂的老面单。

这个设计妙在哪？**各国银行一行旧代码都不用改**。加入 IBAN 体系的成本只是「能生成信封、能拆信封」，而不是迁移几十年的历史数据。IBAN 总长最多 34 位，但每个国家的长度是固定的（德国 22 位、法国 27 位、英国 22 位），机器一看国家代码就知道后面该有多长、该怎么校验。

如今有 80 多个国家用 IBAN，但你可能注意到：美国、加拿大、澳大利亚、中国都不在里面。这不是技术问题——这些国家的国内系统自成一体，跨境业务量相对小，换信封的收益抵不过改造成本。编号系统没有「应不应该」，只有「划不划算」。

编号发出去，就收不回了

IBAN 之所以长得「啰嗦」，根子在于编号的一条铁律：**一旦发出去，就再也收不回**。几十亿个账户号已经印在存折上、写进 ERP 系统、存在客户的收款人列表里，任何一位都不能动。你唯一能做的是包一层、再包一层。

这其实和 IPv4 到 IPv6 的困境（第五章）是同一个病：老地址不会死，新地址只能并存。工程界管这叫「路径依赖」，换个更直白的说法：编号系统的寿命比你想象的长得多，你今天随手定下的格式，五十年后还有人在为它付维护费。SWIFT 自己就是最好的例子——它正在把用了几十年的 MT 报文格式迁到新的 ISO 20022，这场迁移从规划到落地花了二十多年，只因为全世界的银行都得在同一张网里边跑边换轮胎。

所以设计编号时最该问的不是「现在够不够用」，而是「它老了以后怎么办」：留没留扩展位？格式会不会歧义？能不能像 IBAN 那样被套进更大的信封？这一章我们看到的是「编号之间的外交」，下一章要看的更彻底——编号背后的政治。给地球画经纬度的时候，本初子午线放在伦敦格林尼治，可不是天文学家算出来的，是一屋子国家代表投票投出来的。

第十一章 · 给地球编号：经度、纬度与时区

第十一章 · 给地球编号：经度、纬度与时区

前面十章，我们编过快递、编过车、编过钱、编过手机。这一章编一个最大的东西：地球本身。

给地球编号，就是要回答一个问题：**怎么用一个号码说清「球面上任意一点」？**答案你早就知道——东经 116.4°、北纬 39.9° 就是北京。但这两个数字背后，藏着编号设计里最深刻的一课：**一套编号能不能立起来，技术只占一半，另一半是政治。**

纬度和经度：一个白送，一个要命

把地球切成一个坐标系，思路不复杂。先在球面上画两组线：纬度是一圈圈横着切的平行线，赤道是 0°，越往南北两极数字越大；经度是一条条从北极连到南极、竖着切的线，合起来正好 360°——这个数字是从古巴比伦数学借来的，公元前 150 年左右，希腊天文学家喜帕恰斯就这么提议了。

听起来对称、优雅。但实际用起来，这两组线的难度天差地别。

纬度是**白送的**。它有一个大自然给的、无可争议的零点：赤道。水手想知道自己在北纬多少度，晚上看北极星高出地平线多少度就行，一根木杆加一根绳子就能测，误差不大。

经度是**要命的**。大自然没有给经度零点——地球是圆的，任何一条竖线都可以当 0°。而且你在海上根本没法直接测经度：东边的人和西边的人看到的天空一模一样。

不过经度有一个绝妙的间接算法。地球每 24 小时转一圈 360°，所以每小时正好转 15°。这意味着：**时间和经度是同一种东西的两个单位**。如果你随身带着一台钟，钟上永远是出发港的时间；当船上的太阳升到正午（当地时间 12:00），你看一眼钟——如果钟显示下午 3 点，差了 3 小时，你就知道自己在家乡以西 45°。

大白话

翻译成人话：测经度不需要望远镜，需要的是一台在惊涛骇浪里依然走得准的钟。所谓「经度问题」，本质上是个钟表问题。

问题是 18 世纪的钟做不到。钟摆怕晃、齿轮怕锈、润滑油怕冷怕热，船一出海钟就疯走。1707 年，英国一支舰队因为算错经度，在锡利群岛撞上礁石，沉了四艘船，死了近两千人。国会坐不住了，1714 年通过《经度法案》，悬赏**两万英镑**——相当于今天的几百万英镑——给任何能解决问题的人。

最后拿走这笔钱的不是天文学家，而是一个自学成才的木匠钟表匠约翰·哈里森。他花了几十年造出 H4 航海钟，漂洋过海几个月误差只有几十秒。经度问题被一块表解决了——这是编号史上少有的「纯技术就能摆平」的时刻。

本初子午线：22 票对 1 票的妥协

技术解决了「怎么测」，还剩一个更刺手的问题：**0° 画在哪？**

这是纯编号问题里罕见的、大自然不投票的一项。赤道是唯一的，本初子午线（0° 经线）却有无数个候选。于是各国各画各的：法国地图用巴黎，德国用柏林，美国用华盛顿，西班牙用加那利群岛。一条跨大西洋的电报，时间戳在巴黎和伦敦意味着不同的时刻——和第十章里各国银行账号互不认账，是同一种病。

1884 年 10 月，25 个国家的代表被美国总统请到华盛顿开国际子午线会议，任务就一个：投票选出全人类的 0°。结果是 22 票赞成、1 票反对（圣多明各）、2 票弃权（法国和巴西），**格林尼治胜出**——0° 经线从伦敦东南郊的皇家天文台院子里穿过去。

格林尼治赢，不是因为那里有什么地理上的特殊之处，而是因为一个压倒性的现实：**当时全世界 72% 的商船已经在用格林尼治为基准的海图**。这就是编号系统的铁律——占有率就是合法性。设计得再好，不如用的人多。

法国人不服。巴黎天文台的资历一点不差，法国科学院的骄傲不允许 0° 画在伦敦。于是法国弃权，回国继续用巴黎子午线，硬是又撑了 27 年，直到 1911 年才正式改用格林尼治。而且改用时的措辞很鸡贼：不直说「采用格林尼治时间」，而是说「采用巴黎平时减 9 分 21 秒」——面子保住了，里子让了。

大白话

翻译成人话：本初子午线不是被「发现」的，是被「投票投出来」的。编号的零点往往是武断的，真正重要的是所有人同意用同一个。

时区：为什么不按经线直着切

既然每小时 15°，那顺理成章的设计是：把地球竖着切成 24 条，每条 15°，一条一个时区，边界笔直。工程师看了都说好。

但打开真实的世界时区地图，你会看到一堆歪歪扭扭、犬牙交错的边界，像被狗啃过。为什么？

因为**时区的用户不是经线，是人**。一个国家的人希望全国用同一个钟点，方便开会、发车、上班；一个省份不想和隔壁省份差一小时。所以边界在国界和省界面前纷纷拐弯：

- **中国**幅员横跨五个理论时区（从东经 73° 到 135°），但 1949 年起全国统一用北京时间。于是新疆的「上午十点」太阳才刚刚爬上来——编号向行政效率低头。
- **印度**干脆取了个半：UTC+5:30，全国人民折中，谁都不完全满意，谁都不至于造反。
- **尼泊尔**更绝：UTC+5:45。45 分钟的偏移据说是为了不和印度完全一样——连时区都能拿来表达国家个性。

这就是经纬度系统留给我们的最终启示。球面坐标是纯几何，0° 经线是国际政治，时区边界是国内政治。**同一套编号，三层叠下来，越往上越不像数学，越像谈判纪要。**

回顾全书走过的路：快递单号要权衡唯一与可读，IBAN 要向旧系统妥协，本初子午线要靠 22 张选票落地。一条线索反复出现——好编号从来不是算出来的最优解，而是科学、历史与利益掰手腕之后，那个所有人都愿意用的解。下一章，我们把这十一条线索收成一张清单：到底什么样的编号，才配得上「好」字。

第十二章 · 一套好编号长什么样

第十二章 · 一套好编号长什么样

回到第一章那根 15 位的快递单号。当时它只是一串数字；走过十一章之后，它应该已经变成一张可以逐位读懂的图纸了。这一章不做新案例，做一件更实用的事：把前面所有血泪教训压成一份清单。下次你要给自己的系统设计编号——订单号、设备号、API 密钥——可以逐条对照着检查。

第一条：唯一性，但要先问「在什么范围内唯一」

第二章说过，名字崩溃是因为重名。编号的第一使命是唯一，但「唯一」从来不是一个绝对词，它永远带一个作用域。数据库自增 ID 在一张表里唯一，身份证号在一国之内唯一，UUID（通用唯一识别码，一种 128 位的随机编号，不登记、不协商，靠的是「撞到重号的概率比陨石砸中机房还低」）号称在全宇宙唯一。

作用域越大，代价越高。UUID 用 128 位换来了「随便生成都不撞」，代价是那串 `550e8400-e29b-41d4-a716-446655440000` 没有任何人能读。所以清单的第一问不是「要不要唯一」，而是：**这个号码需要在多大的世界里不撞车？**只服务一个小区的快递柜，4 位取件码就够；要服务全球的银行卡，就得老老实实接受 16 位。

第二条：想清楚这个号码是给谁看的

第三章的对照实验还记得吗：UUID 是给机器的，车牌是给人的。给机器看的号码可以丑、可以长、可以毫无意义；给人看的号码必须能念、能抄、能在电话里报给客服。

大白话

翻译成人话：编号设计的第一道岔路口，是「读者是谁」。交警一眼要认出肇事车，所以车牌牺牲了绝对唯一；机器之间握手没人看，所以 UUID 牺牲了人类。两头都想要，结果就是两头都妥协——这正是快递单号又分段又带校验的原因。

第三条：装不下的时候，分段而不是加位

第四章的电话区号、第五章的 IP 和域名，讲的是同一个机制：层级——把一个长号码切成几段，每一段只回答一个更小范围的问题。区号第一段回答「哪个国家」，第二段回答「哪个城市」，最后才轮到具体电话。

层级真正的威力不在「省位数」，而在**分权**：国际电联只管发国家码，发完就不管了；各国自己发城市码，城市自己发局号。没有哪个人需要知道全世界所有电话。一套编号能不能规模化，看的就是它能不能把发号的权力一层层下放出去。第九章的 MAC 地址厂商前缀、第十章的 SWIFT 代码，全是同一个把戏：权威只登记「谁有资格继续往下发」，不登记每一个最终号码。

第四条：假设人会抄错，机器会读脏

第七章的校验位和第八章的二维码，是同一条原则的两代实现：人会抄错，所以编号要会自查。Luhn 算法（信用卡号最后一位的算法，把数字隔位翻倍求和再看能不能被 10 整除）能拦住所有的单个数字抄错、以及绝大多数相邻两位写反——唯一漏网的是把「09」写成「90」这一对特例。后来数学家 Verhoeff 在 1969 年给出了更强的算法，连这类错误也能抓住，但 Luhn 胜在简单到收银员心算都能验，于是活到了今天。

第八章更进一步：与其让人抄完再查错，不如干脆别让人抄。条形码和二维码把编号从「给人看的符号」变成「给光看的图案」，二维码甚至做到缺了三分之一还能扫出来。**容错设计的演进方向，是从「检测错误」走向「消灭抄这个动作」。**

第五条：给未来留座位

编号一旦发出去就收不回，这是第十章的教训：IBAN 之所以长得那么别扭，是因为它要包住几十个国家的旧账号，谁的旧账都不能作废。设计编号时最昂贵的问题不是「现在够不够用」，而是「十年后还够不够」。

ISBN 是个干净的例子。书号原本是 10 位，2007 年 1 月 1 日全球统一切换到 13 位——不是 10 位不够印了，而是要并入商品条码体系，给将来留容量。切换的方式很讲究：所有旧号码前面直接加 `978`，旧号原封不动地活在新号里。对比 IPv4 到 IPv6 那种两套系统并行十几年、至今没切完的狼狈，ISBN 的「向前兼容」堪称教科书。**好的编号系统会给自己留一排空座位，而不是坐满了再拆墙。**

第六条：编号背后必须站着一个管事的

最后一条不在号码里，在号码外。第九章问「谁有权发号」，第十一章的本初子午线告诉你连「零度经线画在哪」都是 1884 年二十几个国家投票投出来的——当时法国投了弃权，因为巴黎子午线落选了。编号系统从来不只是技术设计，它是治理设计：得有一个机构登记、仲裁、处理纠纷，而且这个机构的权威得靠大家认，而不是靠技术强制。

大白话

翻译成人话：技术上完美的编号，没有管事的机构背书，就是一张废纸；反过来，机构再权威，号码设计得没法用，也推不动。经纬度能统治地球两百年，是因为它两头都占了。

拿清单验收：那根快递单号

现在把六条清单拍在第一章的单号上：

- **唯一性**：日期段 + 流水号，保证同一天同一个网点内不撞，跨天靠日期区分——作用域划得很克制。
- **给谁看**：网点编号和校验位给机器，分段结构给分拣员扫一眼就知道去哪。
- **层级**：日期、网点、流水，三段各回答一个更小的问题，发号权下放到网点。
- **容错**：最后有校验位，面单上还有条形码兜底——人手抄和机器扫各有一道防线。
- **演化空间**：15 位对一天的件量是严重超配的，多出来的位数是预留给业务暴涨和扩展新段的。
- **治理**：各家公司自定规则，但电子面单时代行业平台统一了接口，发号权集中到了平台。

六条全中。它不是随便长长的一串数字，而是一份微缩的设计文档。

这本书到这里就讲完了，但编号的故事没讲完。前面所有系统有一个共同的隐含前提：被编号的东西——包裹、电话、银行卡、一本书——数量级都在百亿以内。而下一个正在被编号的世界是物联网：每一颗传感器、每一盏路灯、每一只耳标下的牛都要有自己的号。当编号的对象从「亿」涨到「万亿」，当发号的速度从「每天一批」变成「每秒百万」，这份清单里哪一条会先撑不住？这个问题，留给下一个要设计编号系统的人——也许就是你。

万物皆有编号