Hotel Genius

genius.cat-v.org 蒸馏小册

"Good Evening, I wanted a room between Leonardo da Vinci and Einstein…
Oh, look who's here! Dante Alighieri! How are you?…
Let's have a cup of coffee with Shakespeare…
Dear Newton, how are you?…
Dear Beethoven, do you know Charlie Parker?
Charlie Parker meet Beethoven."
— Roberto Benigni in *Night on Earth* by Jim Jarmusch

Hotel Genius 是 `cat-v.org` 家族里最古怪的一个子站。
它把 Unix 上古的发明者、Lisp 老 Usenet 喷子、自由市场经济学家、
科幻小说家、电影导演、维多利亚剧作家、摇滚怪才——
一起塞进同一家想象中的旅店，互相串门。
本小册把这家旅店十五位住客的房间，逐一蒸馏成一页可读的摘要。

蒸馏自 `genius.cat-v.org` 及其子页面
（视频与音频条目已略去）
2026 年 5 月 15 日

目录

关于这本小册

`genius.cat-v.org` 是 Uriel M. Pereira 主持的 `cat-v.org` 系列站点之一，风格与他在 `harmful.cat-v.org` 上的”xx considered harmful“一脉相承：品味鲜明、毫无中立、专门收录他个人认为”近乎天才“的人物的言论与文字。站点首页用 Jarmusch 电影里那段”想要一间在达芬奇和爱因斯坦之间的房间“做引子，把 Unix 史上的几位创世人物，与喜剧人物、艺术家、经济学家、导演、小说家放在同一个走廊上。

本小册做的事很简单：

- 把站点里 15 个”房间“（每个人物各占一目录）的文字内容逐一蒸馏；
- 补上每个人物的身份信息，方便不熟悉的读者跟得上谈话；
- 抽出每人最有代表性的一两条短语或主张作为脚注；
- 视频、音频、照片之类的条目按用户要求略过。

原站收录的内容覆盖：

- **Bell Labs / Unix** 一脉：Kernighan、Ritchie、Thompson、Pike、McIlroy；
- 软件工程论者：Parnas；
- **Lisp** 与 **Usenet** 上的喷子哲学家：Naggum；
- 科学家与经济学家：Feynman、Friedman；
- 电影与文学：Kubrick、Gibson、Wilde；
- 摇滚与挑衅：Zappa；
- 两个虚构角色：Alex deLarge（《发条橙》）、Eric Cartman（《南方公园》）。

虚构角色之所以与图灵奖得主同登一栏，正是这个站点的”怪“所在：它不是”计算机名人堂“，而是一个有立场的策展。

Chapter 1

Alex deLarge

身份：虚构角色，Anthony Burgess 1962 年小说《A Clockwork Orange》主人公，Stanley Kubrick 1971 年同名电影中由 Malcolm McDowell 出演。
年代：原作设定于不确定的近未来反乌托邦伦敦。
在 **Hotel Genius** 中的角色：原站只给出一句"主要兴趣"作为他的"房间介绍"。

站点摘录

> *"Principal Interests: Rape, ultra-violence and Beethoven."*

蒸馏

这一页只有一行字，却是整个站点的"定调"。它告诉你：Hotel Genius 不是品学兼优名人录，而是一个把贝多芬和暴力并置、拒绝从"艺术能否与道德分离"这个问题里走出来的策展。Alex 这位虚构"住客"在这里的作用是引子——他指向同一家旅店里的另一位住客（Kubrick），并预示了 Burgess 与 Kubrick 共同提出的命题：当人被剥夺善恶选择的能力时，他是否还算是人？（这是 Kubrick 在自己房间里亲口给出的解读，详见后文。）

Chapter 2

Brian Kernighan

全名：Brian Wilson Kernighan（生于 1942 年，加拿大多伦多）。
学历：多伦多大学工程物理本科，普林斯顿电气工程博士（1969）。
履历：1969–2000 在 Bell Labs 计算科学研究中心；2000 年起任普林斯顿计算机科学系教授。
主要贡献：与 Al Aho、Peter Weinberger 共同设计 AWK 的"K"；与 Dennis Ritchie 合著《The C Programming Language》（"K&R"）；与 Rob Pike 合著《The Unix Programming Environment》《The Practice of Programming》；`pic / eqn / ditroff / m4 / ratfor` 等工具的作者或合作者。
Unix 用户名：`bwk`。偏爱的编辑器：sam。

站点收录

原站给 Kernighan 的"房间"是除 Thompson 外最丰富的——有文章、采访、讲座、照片四类。文字部分蒸馏如下。

蒸馏：写程序的哲学

《漂亮的代码》（"**Beautiful Code**" ——**A Regular Expression Matcher**） Kernighan 注解了 Rob Pike 写的 30 行 C 语言正则匹配器（出自 TPOP 第九章），用它来阐释他心目中的"漂亮代码"：

- 简单——易读、清晰；
- 紧凑——只够干活，但不到晦涩；
- 通用——以同一种方式覆盖一大类问题；
- 优雅——透出趣味与品位。

他把 Pike 这段递归实现称作"用 C 语言指针展示递归优雅"的最佳示例之一，并强调：选择正确的特性集（`c . ^ $ *`）本身就是把代码写漂亮的一半。这一篇是 Kernighan 论"工程美学"的浓缩版。

《有时旧办法最好》（**Daily Princetonian，2007**） 关于教学技术的讽刺小品。他作为高科技领域的从业者，公开承认自己在教室里是个”勉强跟随的老派人“：投影仪和黑板能用就好，不需要 PowerPoint，不需要 Blackboard 之外的第二人生。中心一句话：我们要的不是新技术，而是能正常工作的简单老技术。

《奥威尔会怎么做?》（**Daily Princetonian，2008**） 重读《1984》，他指出奥威尔没看到的那种监控：不是国家自上而下的电幕，而是商业的、自下而上的、自愿交出的隐私——随身手机就是”自愿的电幕“，社交网站则是自愿的告白室。隐私会被廉价出售，只要诱惑足够具体。

Princeton 大学公报特写：**Kernighan** 作为教师　描写他放下 Bell Labs 前沿研究、转去本科课堂讲”Computers in Our World“的形象——开领蓝衬衫、坐在金属凳上、把 Mozart 类比为最早的”音乐盗版者“。被描述为”计算机科学里少有的、文字与代码同等被铭记的人“。

多次访谈中的几条主张（综合自 **Linux Journal**、**Computerworld**、**Bell Labs** 等访谈）

- “C 完全是 Dennis Ritchie 的工作。” 他多次强调自己不是 C 的作者，只是合著了那本书。
- Unix 的难学之处不在于命令多，而在于”共享的约定“——像靠右行驶，所有人默认知道，所以从不写下来；外来者撞墙，正是因为这些不成文的惯例。
- Unix 像一个工具箱（toolkit），它从来不是”完成品“——”它做不做 X？做不到，但你想要的话明天就能给你“，这是 70 年代他们的标准对答。

讲座(仅留下文件名） `millions-billions-zillions / the-changing-face-of-programming` ——原站只挂了 PDF 与 MP3，按要求略过音频，主题分别是”大数素养（innumeracy）“和”编程的演变“。

> *“Beautiful code is likely to be simple — clear and easy to understand. Beautiful code is likely to be compact — just enough code to do the job and no more.”*
> — Brian Kernighan

Chapter 3

David Parnas

全名：David Lorge Parnas（生于 1941 年，加拿大）。
学历：卡内基梅隆电气工程博士。
履历：McMaster 大学软件工程教席；曾任职于美国海军研究实验室、IBM 联邦系统部门，为加拿大原子能控制局评审 Darlington 核电站的安全关键软件。
主要贡献：1972 年论文《On the Criteria to Be Used in Decomposing Systems into Modules》，提出信息隐藏（information hiding）原则——今天面向对象编程里”封装“一词的祖先。
公共立场：1980 年代公开反对里根的”星球大战“计划，理由是不可能写出可靠到能阻止核打击的应用软件。

蒸馏

Parnas 在原站只挂了一篇 ACM 的人物访谈，但他在站长心里的位置极高——`harmful.cat-v.org` 的整套”XX considered harmful“传统，其精神祖先就是他这篇 1972 年的论文。

为什么要做模块化？ Parnas 的核心论点听起来朴素：*不要按照流程图把系统切成模块；应当先列出”将来可能改变“或”做起来困难“的设计决定，然后让每个模块各自封住一个这种决定，使其他模块看不见。模块化的目的不是”切得整齐“，而是把变化局部化。*

研究者如何选题 访谈里他反复说：他的研究问题几乎全部来自*工业实践中观察到的现象*，而不是去回应别人的论文。他第一次进入企业是 1969 年带着一个”自以为是的研究想法“，结果在现场发现工程师们不缺”如何写规格“的技巧，而是*从一开始就切错了边界*——所以再精巧的规格也救不回来。他由此转向：与其抱怨规格难写，不如重新切割。

文档作为可数学化的对象 后来的几十年里，他持续做”可被阅读的、近似数学的文档“——用表格表示规格，使非程序员的最终用户也能在草稿阶段找出错误。他对”形式化方法“流派的态度是冷静的：要做的是把已知几十年的数学转换成工程师能用的形式，而不是新发明记号。

> *"…one begins with a list of difficult design decisions or design decisions which are likely to change. Each module is then designed to hide such a decision from the others."*
> — D. L. Parnas, 1972

> *"I would advise students to pay more attention to the fundamental ideas rather than the latest technology.*
> *The technology will be out-of-date before they graduate. Fundamental ideas never get out of date."*
> — D. L. Parnas

Chapter 4

Dennis Ritchie

全名：Dennis MacAlistair Ritchie（1941–2011，美国）。
学历：哈佛物理本科，应用数学博士。
履历：Bell Labs 计算科学研究中心，终其一生。
主要贡献：C 语言之父；与 Ken Thompson 共同创造 Unix；让 Unix 从单机操作系统变成可移植操作系统的推动者；1983 年图灵奖、1998 年美国国家技术奖章。
Unix 用户名：`dmr`，UID 7。偏爱的编辑器：Acme。

蒸馏

Ritchie 自述（2003 年问答） 他给自己最满意的一项工作不是"发明了 C"，而是：*推动 Unix 变成可移植的操作系统*。当年大多数操作系统都是用汇编写、绑死在某一种小型机上的，Unix 因为是高级语言写的，所以可以被搬过去——1970 年代末他与 Steve Johnson 把 Unix 移到 Interdata 机型，正是这一步打破了"硬件—操作系统"绑定的格局。

他对自己的语言判断毫不浪漫：C 在 21 世纪初已经被 C++、Java、脚本语言侵蚀，但"做底层系统的活儿，它还行"。对微内核 vs 宏内核之争，他直接说：实际用起来差不太多。

Rob Pike 对 Ritchie 的悼词（节录意旨） 站长收录的 Pike 悼文，把 Ritchie 的影响概括为：**Unix 是**"大平等者"——它让"软件运行在哪台机器上"变得无关紧要，打破了硬件厂商对编程的统治；当全世界的 mini 机都跑 Unix 时，80 年代的网络发展才能在 Unix 上发生。NeXTSTEP 上跑的是 Unix，TimBerners-Lee 第一个 WWW 服务器也在 NeXT 上——你今天用的网，骨架里仍是 Ritchie 留下的那些约定。

Ritchie 自嘲（来自 `ritchie-on-ritchie`） 被人误把 Unix 的功劳记到 Kernighan 和 Ritchie 名下时，他在 Usenet 写过一封鉴别 K/R/T 三个秃头胡子大佬的"身材鉴定指南"：体型上 Kernighan 最瘦、Ritchie 居中、Thompson 最厚。是站点收录的少数"冷幽默"文本之一。

> *"Unix is very simple, it just needs a genius to understand its simplicity."*
> — 常被归于 Ritchie 的格言

> *“So long, Dennis, and thanks for all the magic.”*
> — Rob Pike，悼 Ritchie（站点收录）

Chapter 5

Doug McIlroy

全名：Malcolm Douglas McIlroy（生于 1932 年，美国）。
学历：康奈尔工程物理本科，MIT 应用数学博士。
履历：1958 年加入 Bell Labs；1965–1986 年任计算科学研究部门主管，即 Thompson、Ritchie、Kernighan 等人的顶头上司；1997 年退休后任 Dartmouth 兼职教授。
主要贡献：发明 Unix 管道（pipes）；写下 `diff` / `spell` / `sort` / `join` / `tr` / `graph` / `speak` 等命令；最早实现实时文字转语音。
Pike 的评价："The unsung hero of Unix"。

蒸馏

McIlroy 在原站的页面短得几乎是个名片，但"名片"里塞着一句关键引用：Rob Pike 把他叫作 Unix 的"无名英雄"。为什么？

管道：唯一一次他动用"主管权力" McIlroy 1964 年的备忘录里写过一句被反复引用的话："We should have some ways of coupling programs like garden hose." ("应该有种办法，能像园艺水管那样把程序串起来。") 他后来自承，把管道塞进 Unix，是他作为部门主管唯一一次动用过的管理权。这一推动让 Unix 在哲学上有了"小程序 + 组合"的根。

被低估的人物 McIlroy 在原站的隐喻分量重于篇幅：他写过 `diff`（让源码控制成为可能）、`spell`（让普通用户在终端就能拼写检查）、`tr`、`sort`——都是 Unix 工具箱里最小、最锋利的那一格。而他主持研究部门的二十多年，让 Thompson、Ritchie、Aho、Kernighan、Pike 等人各做各的，本身就是"管理者的最高德性"的注脚。

> *"We should have some ways of coupling programs like garden hose — screw in another segment when it becomes necessary to massage data in another way."*
> — Doug McIlroy, 1964

Chapter 6

Eric Cartman

身份：虚构角色，Trey Parker 与 Matt Stone 的动画剧集《South Park》（南方公园，1997 至今）四主角之一。
家乡：South Park, Colorado。
标签：MSNBC 曾评其为电视上”第二可怕的角色“。

站点摘录

> *“Respect mah authoritah!”*
> *“Sweeet!”*

原站还收录了 NPR 的”Proust 问卷“访谈片段（音频），脚本记录里只有这一段开场：

Julie：*“Eric,* 我想问你几个问题，让听众更了解你，这是 *Proust* 推广、*James Lipton* 发扬过的问卷。
　　你听说过他们任何一个吗？ ”
Eric：*“Eh, no, I don’t really care about that.”*

蒸馏

Cartman 与 Alex deLarge 一起，是 Hotel Genius 里的”虚构住客“，功能是自嘲与平衡：在一个充满 Turing 奖得主与诺贝尔经济学奖得主的旅店里，站长偏偏给一个胖小学生留了一间房——他对 Proust 问卷的回答是”我不在乎“。这是站点品位的一部分：拒绝精英姿态、拒绝庄严，并把这份拒绝本身也作为一种态度展示出来。

Chapter 7

Erik Naggum

全名：Erik Naggum（1965–2009，挪威奥斯陆）。
履历：Naggum Software 创办人；为 Emacs 贡献近十年；RFC 1123（互联网主机软件要求）与 RFC 2049（MIME）合作者之一；长年在 `comp.lang.lisp` 等 Usenet 群组上活跃。
身故：44 岁因溃疡性结肠炎导致的大出血去世。
江湖名号：传奇 Lisp 程序员、Usenet 战神（flame warrior）；对 Perl 和 C++ 有近乎宗教式的反感。

蒸馏

Naggum 不出书、不开公司讲技术博客，他留下的几乎全是 **Usenet** 帖子。genius.cat-v.org 把其中五段精选下来当作”房间“：

《Lisp Markup》《XML–SGML–NML–Lisp》《SGML 属性与元数据》（2001 反复出现的主题） 他对 XML / SGML 的根本批评是：”元素“和”属性“之间那条 *syntactic* 边界纯粹是历史惯性，没有任何语义必要。SGML 把容器内容和”盒子上写的字“分两种语法表达，造成了大量重复机制；而 Lisp 的 S-表达式从一开始就只有一种容器写法，让结构（content）与元结构（metadata）可以无缝嵌入。他在与 SGML 之父 Charles Goldfarb 合作多年之后，得出结论：那是个根本性的设计错误。

《Perl Rant》 站点最常被引用的一篇。Naggum 把那个”想找工作所以打算学 Perl 的程序员“形容为：原本只有一个问题（没工作），现在有两个问题（没工作 + 学了 Perl）。他承认自己曾把 Perl 啃到能看懂代码、能在别人程序里找出 bug——但他后来意识到，这”不算什么成就：在 Perl 程序里发现 bug，就像发现一条狗身上带着跳蚤。“

《惩罚者与道德家》（Punishers and Moralists） 2008 年他与 Tobias Rittweiler 的私人通信。Naggum 描述自己长年与 Usenet 上”惩罚者“相处的代价——这些人本身什么都生产不出来，但执着于让别人为没活成他们规定的方式付出代价。站长引用他的座右铭：

> *“Some people are little more than herd animals, flocking together whenever the world becomes uncomfortable…*
> *I am not one of those people.*
> *If I had a motto, it would probably be: Herd thither, me hither.”*
> — Erik Naggum

为什么 **Hotel Genius** 给他留了房间　他是站点”修辞英雄“的化身：在一种公开、敌意、毫无礼貌的媒体上，长篇大论地、用词精确地讲技术与哲学。他与 Wilde、Zappa 一起，构成了这家旅店里”毒舌艺术家“的一翼。

Chapter 8

Frank Zappa

全名：Frank Vincent Zappa（1940–1993，美国巴尔的摩）。
身份：作曲家、吉他手、乐队 Mothers of Invention 的核心；横跨摇滚、爵士、当代古典、电子；1985 年因反对 PMRC（家长音乐资源中心）唱片标签法案在美国参议院作证。
产出：生前发行约 60 张专辑，身后整理出来的还在继续增长。

站点摘录

> *Information is not knowledge.*
> *Knowledge is not wisdom.*
> *Wisdom is not truth.*
> *Truth is not beauty.*
> *Beauty is not love.*
> *Love is not music.*
> ***Music is the best.***
>
> — Frank Zappa（出自 *Packard Goose*，《Joe's Garage》Act III, 1979）

蒸馏

Zappa 的房间里就只挂了这一段七行的"信息梯子"。它的形式像逻辑命题，内容却是反逻辑的：每一步都断开两个常被并置的词，最后落到"音乐才是最好的"——这是 Zappa 一贯的反讽态度：拒绝把音乐当作思想的载体，拒绝把"艺术服务于真理与美"的浪漫主义当作不证自明的前提。

站点收录他正是因为这种立场与 `cat-v.org` 整体的"不庄严但有标准"合拍。

Chapter 9

Ken Thompson

全名：Kenneth Lane Thompson（生于 1943 年，美国）。
学历：UC Berkeley 电气工程本科与硕士。
履历：1966 加入 Bell Labs；2006 加入 Google。
主要贡献：Unix 的主要发明者；B 语言、Go 语言的设计者之一；正则表达式在程序中的最早实现者；写过国际象棋程序 Belle（与 Joe Condon）；1983 图灵奖（与 Ritchie 同享）、1999 国家技术奖章。
Unix 用户名：`ken`，UID 6。偏爱的编辑器：sam。

蒸馏

Thompson 的房间是站点最丰富的——访谈、引语、文章、迁居加拿大（“mig”）小相册。

Unix 的来历（1989 访谈，与 Mahoney 谈话） 他自承 Unix 不是凭空冒出来的：来自他参与过的三个系统的”最干净那一份“——Berkeley 的 SDS940（Project Genie）、MIT 的 CTSS、以及失败但有意思的 Multics。管道这种”把文件和设备等同看待“的想法，当时业界都觉得应该这么干，但谁都没真做出来；他做了。另一个”近乎宗教“的决定：从一开始就坚持要用高级语言写操作系统。他说，“Right from the start. Knew we had to.”

怎么做出强大的抽象？（IEEE Computer 访谈） 他自称是*自下而上的思考者*——你给他对的乐高积木，他能看见半英里高的楼；但你拿来一个层层封装的复杂系统让他看，他就只能看到一团雾。他不无幽默地猜测：自己之所以做出 Unix 这种”极简又极强“的东西，也许只是因为他造不出更复杂的东西自己还能理解。

Reflections on Trusting Trust（1984 图灵奖演讲） 这一篇是他亲笔写下的最著名文本，站点单独存了一份。全文示范了一种著名的安全攻击：你可以让编译器里有一段后门，即使你给别人看完整的源码，那段后门也能在编译过程中*自我复制*回去。结论：*你不能信任不是你自己亲手写的代码*——*甚至包括编译器写的代码*。这是计算机安全里被引用最多的论文之一。

站点收录的两段引语

> *"Even so, if the way is too radical, no one will follow it.*
> *Every important decision was weighed carefully.*
> *Throughout,* ***simplicity has been substituted for efficiency****.*
> *Complex algorithms are used only if their complexity can be localized."*
> — Ken Thompson, *UNIX Implementation*, 1978

> *"Our collaboration [with Dennis Ritchie] has been a thing of beauty.*
> *In the ten years that we have worked together, I can recall only one case of miscoordination of work.*
> *On that occasion, I discovered that we both had written the same 20-line assembly language program.*
> *I compared the sources and was astounded to find that they matched character-for-character."*
> — Ken Thompson, *Reflections on Trusting Trust*, 1984

Chapter 10

Milton Friedman

全名：Milton Friedman（1912–2006，美国）。
学历：芝加哥大学经济学硕士、哥伦比亚博士。
履历：芝加哥大学经济系；Hoover Institution 高级研究员；1976 年诺贝尔经济学奖。
主要贡献：货币主义、宏观经济史（与 Anna Schwartz 合著《A Monetary History of the United States》）、对凯恩斯主义的系统性挑战；公共形象上是 20 世纪后半叶最有影响力的自由市场倡导者之一。
大众作品：《Capitalism and Freedom》《Free to Choose》（同名公共电视系列）。

蒸馏

原站收录了他两次访谈。

《The Region》访谈（明尼阿波利斯联储，1992） 要点：

- 反对凯恩斯主义式财政刺激：所谓"增加政府支出以刺激增长"的呼吁，在他看来既不政治可取（"临时支出最容易变成永久"），也不技术上成立（财政赤字只有在货币扩张配合下才有刺激效果）；即使要刺激，减税也比增支更可取，因为减税不会进一步扩大政府规模。
- 对存款保险态度的转变：他与 Schwartz 在 1930 年代背景下肯定 FDIC——因为美联储当时未能履行使命；但 1970 年代的通胀摧毁了储贷机构的净资产，*在没有了"所有者本人的钱在博弈"之后*，存款保险变成了赌徒的"稳赚不赔"，因此后期他支持改革。
- 对欧元的预测：他不认为欧洲货币联盟在他有生之年（也许在采访者有生之年也未必）能真正实现单一货币。一个统一货币需要：劳动与商品自由流动、利益足够同质、单一中央银行。这三条欧洲都不具备。

New River Media 访谈（" Commanding Heights "系列） 要点：

- 大萧条不是 **1929** 年股灾，而是 **1929–33** 年货币崩塌：美联储让货币存量缩水近三分之一，*"这是它本来被设立出来要防止的事，结果反倒把它放任、加剧了。"*
- 关于知识与权力：他认为 20 世纪美国公众对"政府能不能办事"的态度转变，不是被某一篇文章说服的，而是被经验教训的——反贫困战争后贫困加剧、福利项目膨胀、邮局成笑话。

但思想史的作用在于：当经验导致危机、人们准备改变时，必须已经有一套替代方案在手上。这是他对"长期不动摇地讲少数派立场"的辩护。

> *"You never have real changes unless you have a time of crisis.*
> *And when you have a time of crisis what happens depends on what ideas are floating around, and what ideas have been developed, and thought through, and are made effective."*
> — Milton Friedman

Chapter 11

Oscar Wilde

全名：Oscar Fingal O'Flahertie Wills Wilde（1854–1900，爱尔兰都柏林）。
身份：剧作家、诗人、小品作家；代表作《The Importance of Being Earnest》《The Picture of Dorian Gray》《De Profundis》；1895 年因"严重不雅"罪被判劳役两年，1900 年贫病死于巴黎。

站点收录

genius.cat-v.org 给 Wilde 留的房间，正中央只摆了一篇文本：《道林 · 格雷的画像》开篇的序言（Preface），全文不到一页。

蒸馏

这篇序言是 Wilde 自己为应对 *Dorian Gray* 出版后遭遇的道德围攻写下的"艺术宣言"。要点：

- 艺术家创造美的事物，作品中应当隐藏艺术家本人；
- 在美的事物里找到丑意的人"corrupt without being charming"，是缺陷；找到美意的人则是被启蒙过的人；
- "There is no such thing as a moral or an immoral book.
  Books are well written, or badly written. That is all."
- 评判艺术对"意见多样性"的反应：意见相左恰好证明作品是新的、复杂的、有生命的；
- **All art is quite useless.**

Wilde 站在 Hotel Genius 走廊里的作用，是给整本小册的策展原则提供一个出处——以美学判断代替道德判断，以"做得好不好"代替"应不应该做"。这是站点的底层语法。

> *"All art is at once surface and symbol.*
> *Those who go beneath the surface do so at their peril.*
> *Those who read the symbol do so at their peril.*

> *It is the spectator, and not life, that art really mirrors."*
> — Oscar Wilde, Preface to *The Picture of Dorian Gray*, 1891

Chapter 12

Richard Feynman

全名：Richard Phillips Feynman（1918–1988，美国纽约）。
学历：MIT 本科、普林斯顿博士。
履历：曼哈顿计划成员；康奈尔、Caltech 教授；1965 年与 Tomonaga、Schwinger 同享诺贝尔物理奖（量子电动力学）；1986 年挑战者号航天飞机事故调查委员会成员。

蒸馏

原站收录了他的两封信、一份演讲、一篇毕业致辞、一份政府报告附录。五份文本贯穿同一个 Feynman：对自欺与浮夸保持彻底警惕的科学家。

《There's Plenty of Room at the Bottom》（1959, 美国物理学会年会演讲） 被公认为纳米技术之父的演讲。他用一个具体的算术展示"有多少空间"：若把一根针头放大 25 000 倍，其面积等于整本《不列颠百科全书》全部书页之和。所以原则上完全可以把整部百科写在一根针头上——而当时已有的成像技术足以阅读它。他没有发明纳米技术，但他指出了"底下还有大量未做的工作"这件事的工程可能性。

《What Problems to Solve》（致 Koichi Mano 的信） 他的学生写信抱怨自己被分配到"湍流大气中电磁波传播"这种"低端、谦卑"的问题。Feynman 回信批评的是这种思维："有价值的问题，是你能真的解决、能真的为之贡献的问题。"真正令人愉悦的是把简单问题解干净，"哪怕只是回答了一个比你笨一点的同事心里的疑问。"

《You Don't Understand "Ordinary People"》（致 Stephen Wolfram 的信，1985） Wolfram 向 Feynman 提议要建一个"复杂性研究所"，由自己主持。Feynman 在回信里清醒地点破：你想造一个你能干活的环境，但你不会真的在那里干活——你会去管理它。而你恨管理，你不擅长。更扎心的一句：你不懂"普通人"。在你眼里他们是"蠢瓜"，所以你不会容忍他们，于是你也不会成功地管他们。

《Cargo Cult Science》（1974 Caltech 毕业演讲） 人类用"尝试 → 留下管用的 → 丢掉不管用的"这套方法发展出了科学。但他抱怨现在依然到处是"货物崇拜科学"——表面有所有形

态，却缺少最关键的诚实：”对自己讲真话——自己是**最容易骗的人**。“科学家应当主动把可能让自己结论被推翻的细节先列出来给读者；广告、政府、伪科学都正相反。

《On the Reliability of the Shuttle》（挑战者号事故报告 **Appendix F**） 对 NASA 的批评直接得近乎残酷：工作工程师认为整机失败率约 1%，管理层认为是 1/100 000——”那就意味着每天发射一架航天飞机连续 *300* 年才会损失一架。“他追问：管理层这种近乎魔法的信心从哪里来？报告把”发射阅检准则随时间逐渐放宽“、”同一种风险因为上次没炸所以这次也能接受“这种制度疾病记录在案。最后那句他自己说出来的话——”*For a successful technology, reality must take precedence over public relations, for nature cannot be fooled.*“——后来反复被引用到所有”技术与组织行为“的讨论里。

> *“The first principle is that you must not fool yourself — and you are the easiest person to fool.”*
> — Richard Feynman, *Cargo Cult Science*, 1974

> *“For a successful technology, reality must take precedence over public relations, for nature cannot be fooled.”*
> — Richard Feynman, Challenger Report, Appendix F, 1986

Chapter 13

Rob Pike

全名：Rob Pike（生于 1956 年，加拿大多伦多）。
履历：Bell Labs、Google；与 Ken Thompson 在 Google 共同设计 Go 语言。
主要贡献：Plan 9 与 Inferno 操作系统主要作者之一；编辑器 `jim`、`sam`、`acme`；窗口系统 `rio`；和 Thompson 一起发明 UTF-8；合著《The Unix Programming Environment》《The Practice of Programming》；1985 年《Program Design in the UNIX Environment》（即著名的 `cat -v` Considered Harmful 论文）。

蒸馏

Pike 的房间里有论文、文章、相册、还有一份芝士蛋糕食谱。

《Systems Software Research is Irrelevant》（Utah 2000，亦称 utah2k） 他著名的”系统研究已死“演讲：操作系统的生态位已经填满，新的好点子越来越难提出、更难传播。Ritchie 也基本同意他这个判断，只是嫌他说得太挑衅了一点。

《Notes on Programming in C》 他给”如何写 C 才不令人讨厌“留下的内部风格指南，精神和 Kernighan/Plauger 的《The Elements of Programming Style》是一脉的：程序的清晰本身比花哨重要；命名、缩进、控制流要服从于让下一位读者活下去。

《UTF-8 History》 他和 Ken Thompson 在新泽西一间小餐馆 Murray Hill 的纸垫子上发明了 UTF-8 编码，然后用一周左右时间把 Plan 9 全系统切换过去。这是他们留给现代互联网的最大基础设施之一：ASCII 兼容、字节自同步、变长高效。今天你看到的几乎每一个网页、每一条 JSON，都跑在这个”一周做出来的东西“之上。

《Cheesecake I》 Pike 1986 年在 Usenet `mod.recipes` 上发的”传奇芝士蛋糕“食谱，来自一位”做甜品的朋友的朋友“，据说拿过奖。他自评是”他做来给新朋友吃，第一次吃的人多半会脱口而出这是我吃过最好的芝士蛋糕“的那种。重点是：所有食材都要室温（用冷奶油芝士保证会出疙瘩），打芝士时慢慢搅、勤刮碗壁，蛋一颗一颗加。难度评级：”相当难（火候要紧）。“站

点把它郑重其事地放进 “Rob Pike / texts/” 而不是 “photos/”，是一种 cat-v 式的幽默——一个写过 *UTF-8* 论文的人也写过芝士蛋糕配方，两者都收进文本目录。

> *“Data dominates.*
> *If you’ve chosen the right data structures and organized things well,*
> *the algorithms will almost always be self-evident.*
> *Data structures, not algorithms, are central to programming.”*
> — Rob Pike, *Notes on Programming in C*（站点收录的指南之一）

Chapter 14

Stanley Kubrick

全名：Stanley Kubrick（1928–1999，美国纽约）。
身份：摄影师、电影导演、剪辑师、制片人。
代表作：*Paths of Glory*（1957）、*Spartacus*（1960）、*Lolita*（1962）、*Dr. Strangelove*（1964）、*2001: A Space Odyssey*（1968）、*A Clockwork Orange*（1971）、*Barry Lyndon*（1975）、*The Shining*（1980）、*Full Metal Jacket*（1987）、*Eyes Wide Shut*（1999）。

蒸馏

原站收录了五份重要文字访谈：与 Michel Ciment 谈三部片（*Clockwork Orange / Barry Lyndon / The Shining*）、1968 Playboy、1987 Rolling Stone。Bernstein 在《纽约客》的早期访谈以 mp3 形式存档（略）。

《A Clockwork Orange》（与 Ciment） 他自己给电影定调：核心议题是自由意志——"若被剥夺善与恶的选择，我们是否还是人？" Alex 经过 Ludovico 疗法被"文明化"，随之而来的病也许就是社会施加给自然人的神经症。他不接受影片里那个反乌托邦是"共产主义"的解读：极右的部长和极左的作家在剧中代表的是同一种东西——"他们的手段与目的几乎无从区分，只是教条不同。"对"科学家比国家更糟"的指控他也予以澄清：真正危险的是科学发展出毁灭手段的速度超过了我们学会使用它们的速度，但责怪科学本身愚蠢——科学不能被控制，因为没人有资格控制它。

《Barry Lyndon》（与 Ciment） 他选 Thackeray 这本"反英雄"小说，正是因为它和当时观众期待的"有内心成长的主人公"相反——Barry 不是一个被读者认同的角色，他只是被时代推着走的浮浪子。Kubrick 把整部电影拍得像 18 世纪英国与欧洲风景画：长镜头、自然光（包括著名的 NASA 蔡司 f/0.7 镜头拍烛光场景），画意先于剧情。这正是他作为前 *Look* 杂志摄影师的本能延续。

《The Shining》（与 Ciment） 他强调这不是一个"寓言电影"：他无意把 Overlook 旅店当成美国种族屠杀史的隐喻——观众想看到什么，会自己看到。他真正在意的是鬼故事这种类型本身："鬼故事是少数几种可以让观众进入电影里去当那个角色的题材之一——你愿意被吓，所以

你借给它一份信任。“他承认 Jack 的家庭暴力倾向在故事开始之前就在，旅店做的只是放大，不是创造。

Playboy 1968 与 Rolling Stone 1987 两次访谈相隔近 20 年，关心的事却高度一致：

- 电影的本质是节奏与图像，不是台词；
- 不信任”心理学“式的人物动机解释——更愿意像作家那样把人物推进矛盾里看他们怎么动；
- 对宗教与超越问题（2001 末尾的”星孩“、Shining 里的”那个旅店“）保持非教条式的开放；
- 对工业体系（《Spartacus》失败的体验）有持久戒心，因此从 60 年代后期起几乎完全自制自剪自控母带。

> *“The central idea of the film has to do with the question of free-will. Do we lose our humanity if we are deprived of the choice between good and evil?”*
> — Stanley Kubrick on *A Clockwork Orange*

Chapter 15

William Gibson

全名：William Ford Gibson（生于 1948 年，美国南卡罗来纳，长居加拿大温哥华）。
身份：科幻小说家，*cyberpunk*（赛博朋克）一词与流派的发明者之一。
代表作：*Neuromancer*（1984，开 cyberspace 之先河）、*Count Zero*（1986）、*Mona Lisa Overdrive*（1988）、*Pattern Recognition*（2003）、*Spook Country*（2007）、*The Peripheral*（2014）。

蒸馏

站点摘录（开场白）

> *"Well-intentioned people who invite me to 'brainstorm' about serious Future Stuff don't really grasp my function. Which isn't their fault."*

这一句奠定了他在站点里的角色：他不是"未来学家"，他自承自己是从已经在场的当下里看出"未来已经在了"的人。

《Google's Earth》（2010 New York Times 评论文章） 对 Eric Schmidt"用户其实希望 Google 告诉他们下一步该做什么"的那句争议话，Gibson 给了一个温和但毫不浪漫的回答：我们确实希望。我们幻想过 HAL 9000 那种"瓶子里的精灵"，但 Google 不是那种东西——它是分布式的双向膜，是我们一同织出来的"珊瑚礁"：我们每一次搜索都是免费内容，集合起来变成它。他指出更深的一层："赛博空间已经外翻（*everted*），把自己翻成了表面，殖民了物理世界。"我们不再"进入网络"——网络已经是我们行走其上的地面。那它就不再是 Bentham 的圆形监狱（中心一只看守的眼睛），而是一种每个人都同时是被监视者和监视者的视网膜细胞的状态。

Larry McCaffery 访谈（1986） *Neuromancer* 出版后不久的对谈。Gibson 解释 cyberspace 这个词的来源：他听到"电子游戏厅里小孩盯着屏幕的眼神"与"晶体管广告里的格子图"在脑子里融合，"那一刻我意识到屏幕背后应该有一个空间。"他坚决否认自己懂技术——他写《Neuromancer》时根本没用过电脑，是在一台便宜的 1927 年款 Hermes 打字机上敲完的。这是他作为科幻作家的底色：未来在感觉里，不在硬件目录里。

《Digital Culture》访谈（1994） 他在 90 年代中期已经在警告"互联网最终会变得跟 1950 年代电视一样规驯"的可能；而他对早期 Web 的兴奋点也很 cyberpunk 式：真正有意思的不是技术规格，而是各种亚文化和黑市在它边缘自发长出来的样子。

《Irish Times》访谈（2003，*Pattern Recognition* 出版前后） 他正式从"近未来科幻"转向"同时代小说"的时点。理由是："我们眼下的现实已经足够 *sci-fi*，写未来反而过时。""*The future is already here — it's just not very evenly distributed.*"这一句虽不出自本访谈，却是他这一时期反复对各家媒体讲的同一个意思的格言版。

> *"Cyberspace, not so long ago, was a specific elsewhere…*
> *Now cyberspace has everted. Turned itself inside out.*
> *Colonized the physical."*
> — William Gibson, *Google's Earth*, 2010

尾声：这家旅店为什么这样配房

把十五个房间走一遍，可以看出策展人 Uriel 给 Hotel Genius 的几条隐藏标准：

标准一 ”简单替代效率“的偏执。Thompson 在 1978 年的 Unix 实现报告里那句”simplicity has been substituted for efficiency“，Kernighan 论 “beautiful code” 的紧凑标准，Parnas 论模块化时的”把决策局部化“，McIlroy 论管道时的”像水管一样接起来“——几乎是同一种品位的不同发音。

标准二 对自我欺骗的警惕。Feynman 的”first principle is that you must not fool yourself“、Friedman 对凯恩斯主义自欺的清算、Kubrick 对”艺术与政治都偏好极端“的冷眼、Naggum 对 Perl/XML 的”是你们自己骗自己“的怒火——在这家旅店里被反复使用。

标准三 毒舌权。Wilde 给了序言式的辩护，Zappa 给了反讽式的口号，Naggum 给了 Usenet 式的实战示范，Cartman 给了反精英的滑稽镜像——Hotel Genius 不要求住客谦逊，只要求他们准确。

标准四 虚构与真实并置。Alex 与 Cartman 与图灵奖得主同住一层——因为站点显然认为，对人的判断力，一段贴切的台词与一篇论文同样有效。

如果有第十六个房间，按这套品位猜，应当留给一位毒舌的、对自欺零容忍的、文字精确到病态的、又愿意写芝士蛋糕食谱的人。读者诸君，自行提名。

— fin —

出处与说明

本小册的”站点摘录“与各人物的页面结构，全部蒸馏自 `genius.cat-v.org` 及其子页面（截至 2026 年 5 月抓取，原站使用 *werc* 框架）。传记信息综合自 Wikipedia、ACM、Bell Labs、Caltech、Dartmouth、Princeton CS、IMDb 等公开资料。

原站收录的视频（`youtube embed`）与音频（`mp3`）按用户要求未蒸馏。

排版风格沿用 `Bell_Labs_cat-v.tex` 与 `Urial-guide.tex`，以 ctexbook + xelatex 编译。