碼語辭典

代码里的汉语

中文开发术语的语言学考

> 「单个字都认得，连起来却像暗号。」

共 235 则词条　八辑 · 一则附录　八种造词法　Fable 撰

序言：把「边车」和「架构首付」当成一门方言

你大概注意到了：程序员——包括跟你聊天的 AI——嘴里的中文，简短、精确，还总蹦出一些日常见不到的词。「静默消失」「边缘崩溃」「架构首付」「边车」「冒烟通过」……单个字都认得，连起来却像暗号。

这不是错觉。开发汉语是一门不折不扣的**行业社会方言**（technical sociolect）：一个说汉语的技术社群，在半个多世纪里，为了给一批本无中文名字的概念命名，同时动用了三套资源——**与英文行话的翻译接触**、**文言的构词传统**、以及**网络与市井口语**。三股力量在同一个词上打架又和解，就长出了这些「眼熟又陌生」的词。

这本小辞典，是给这门方言做的一次语言学速写：两百多个高频词，按开发流程分成八辑。每个词不只告诉你它是什么意思，更要拆给你看——它是怎么造出来的，为什么生动，又为什么难懂。

八种造词法

看多了你会发现，这门方言反复在用同样几副模具。理解了模具，陌生词就能自己拆解。

1. **仿译（借译，calque）**——把英文的比喻逐字直译过来。例：边车（sidecar）、心跳（heartbeat）、金丝雀、看门狗、墓碑、面条代码。麻烦在于：搬来了比喻，却常常没搬来那幅画面。
2. **意译优先，音译罕见**——取义不取音，几乎不像「沙发」「咖啡」那样照读。例：幂等、句柄、门面。翻遍全册也找不出几个音译词，这是汉语技术翻译的鲜明脾气。
3. **四字并列格（文言骈俪）**——用成语式的对仗压缩一整套策略。例：削峰填谷、读写分离、分库分表、纵深防御。四个字里能藏一页设计文档。
4. **动宾／动补紧缩**——单音节动词配一个宾语或补语，密度极高。例：打桩、落盘、回源、探活、埋点、提测、双写。省掉的从来不是信息，而是「默认你已经知道」的共识。
5. **隐喻引申**——把日常、自然、建筑、军事、身体的意象搬进技术。例：兜底、雪崩、脑裂、屎山、堡垒机。
6. **语义窄化**——一个常用词被行业征用后，词义收窄、甚至翻面。例：消费（一条消息）、快照、对齐、脏（页）、降级。日常义常常帮倒忙。
7. **字母词与语码转换**——拉丁字母缩写直接嵌进汉语句法，还能屈折活用。例：PR、CI、QPS、OOM、P99；「mock 掉」「RAG 一下」「rebase 一下」。
8. **语域反差与感情色彩**——文言雅词与粗俗黑话在同一行当里共处，褒贬还常常倒挂。例：文绉绉的幂等、契约 ↔ 骂人的屎山、甩锅；名褒实贬的乐观锁、优雅降级、漏桶。

一条逆流：中文反向输出

多数词是英文流入中文，但也有几个方向相反——概念在中国的工程实践里长出来，英文只能回译。例：中台、埋点、提测、脱敏。它们背后往往不是一个概念，而是一套**制度**（开发／测试的正式交接、全链路压测的组织现实），值得被专门命名。

为什么初学者看不懂

同一个难点，往往落在五个地方：

- **喻体隐身**：仿译搬来了比喻却丢了画面——「边车」不提摩托挎斗，「银弹」不讲狼人。
- **字面与义项脱节**：语义窄化后日常义误导人——「回归测试」的回归是坏事，「乐观锁」根本没有锁。
- **四字格压缩**：一个成语状的词里塞了半篇文档——「削峰填谷」不告诉你怎么削。
- **字母词无提示**：PR、P99、OOM 直接上句，不解开缩写就寸步难行。
- **语域落差**：同一件事，事故报告写「宕机」，工位上喊「挂了」，书口两套词，得两套都会。

凡例

每个词条包含以下几栏：

- **词头**：术语本身，用「／」列出常见同义变体。
- **英文源 · 拼音**：概念的英文出处（若为本土原创会注明），及带声调的普通话拼音。
- **字面**：逐字的本义；若是隐喻或仿译，点明喻体来自哪个现实场景。
- **释义**：它在开发语境中的精确含义。
- **例句**：一句真实、地道的开发对话或文档用语。
- **语言学注**：全书的重心。归纳它用了上面哪种造词法，属于什么语域、带什么感情色彩，以及它对新人为何难、对老手为何顺。
- **关联**：可参看的相关词条。

第一辑 · 故障与可靠性

系统如何失效，又如何被教会体面地失效——从「静默消失」到「兜底」。

静默失败／静默消失

silent failure · *jìng mò shī bài*

字　面　静默＝不出声。失败发生得无声无息。

释　义　程序出错却不抛异常、不打日志、不给任何提示，错误被吞掉、数据悄然丢失（静默消失），系统表面一切正常。

例　句　「这层 `catch` 把异常吞了还返回默认值，线上跑成了静默失败，数据错了三天才有人发现。」

> **语言学注**
>
> 仿译 silent failure，但真正的机关在预设：它先把「报错」隐喻为「出声」，程序被拟人成该喊疼的生命体——不点破这层隐喻引申，newcomer 就无法理解「安静」为何成了罪名（fail fast 的价值观即「大声失败」）。「静默」是书面雅词（静默哀悼），与口头骂法「异常被吃了」构成语域反差：同一现象，事故报告用前者，工位吐槽用后者。强贬义，工程上公认最恶劣的失败方式。

关　联　兜底、假阳性／假阴性、副作用

边缘情况／边界情况

edge case / corner case · *biān yuán qíng kuàng*

字　面　处在「边缘／边界」上的情形。喻体来自几何：把输入取值范围想成一块平面，常规输入在腹地，极端输入在边上。

释　义　输入或状态处于取值范围极端、或多个极端交汇处，主流程覆盖不到、最容易出错的情形。

例　句　「主链路的 case 都过了，就剩几个边缘情况：空列表、恰好跨零点、还有闰年二月二十九。」

> **语言学注**
>
> 仿译 edge case。英文其实分两档：edge case（单一参数到边）与 corner case（多个极端在「墙角」相撞，更刁钻），汉译把两者合并为「边界／边缘情况」，是一次悄悄的语义合并损耗——中文使用者从词面上感知不到这个精度差。空间隐喻引申是理解关键：newcomer 常误以为「边」指界面边缘或业务边界，不知道它指参数空间的几何边界。语域中性、测试评审高频。

关　联　长尾场景、假阳性／假阴性、兜底

长尾场景

`long-tail scenario` · *cháng wěi chǎng jǐng*

字　面　长长的尾巴。喻体来自统计图形：把场景按出现频次排序，曲线右侧拖出一条细而长的尾。

释　义　单个发生概率极低、但种类繁多、加起来不可忽略的罕见场景集合。

例　句　「自动驾驶难的不是主路巡航，是长尾场景——逆行的三轮车、掉在快车道中间的沙发。」

> **语言学注**
>
> 源头是统计学的幂律「long tail」，经《长尾理论》商业畅销书泛化进大众语汇，再被工程界借回来——一条「学术→商业→技术」的词汇回流路线。属仿译，且是典型的图形隐喻：不在脑中画出频次—排名曲线，就不知道「尾」在哪、为何「长」，这正是 newcomer 的门槛。与「边缘情况」侧重不同：边缘讲取值极端，长尾讲「个个罕见、合计常见」的统计结构，老手用它一词点出「测试永远测不完」的残酷事实。

关　联　边缘情况、长尾延迟、兜底

竞态条件／竞态

`race condition` · *jìng tài tiáo jiàn*

字　面　竞＝竞争、赛跑；态＝状态。结果由「谁跑得快」决定的状况。

释　义　多个线程或进程的执行时序不确定，程序结果依赖「谁先到」，导致偶发、难复现的错误。

例　句　「这个 bug 只在压测时冒头，十有八九是竞态——两个线程同时读改写同一个计数器。」

> **语言学注**
>
> 仿译 race condition，却带着一处经典误配：英文 condition 此处是「状况」义（如医学的 condition＝病况），而汉语「条件」的主义项是「前提要求」，于是「竞态条件」字面读起来像「发生竞争的前提」，初学者往往在错的义项上打转。更地道的汉语操作是把它紧缩成双音节「竞态」，再当类词缀繁殖：竞态窗口、数据竞态——体现汉语双音化的压缩力。另一重难点是施事隐身：谁在竞、竞什么（CPU 调度权）字面全无提示。贬义，bug 归因语境高频。

关　联　死锁、活锁、幂等、副作用

死锁

`deadlock` · *sǐ suǒ*

字　面　死＝解不开的、彻底的；锁＝锁具。锁死了，谁也打不开。

释　义　两个以上执行体各自握着对方需要的资源互相等待，全体永久卡死，无外力介入不能恢复。

例　句　「两个事务互相等对方释放行锁，InnoDB 检测到死锁，直接回滚了代价小的那个。」

> **语言学注**
>
> 教科书级仿译：dead→死、lock→锁，语素一一对应。它落地毫无违和，是因为汉语「死」作前缀本有「固定、不能变通」的能产义——死结、死胡同、死记硬背——与 dead- 的「彻底、不可逆」义天然重叠，借形借义两头都接得上，以致母语者几乎感觉不到这是外来词。对 newcomer 反而友好：字面即义，还能口语化为「互相锁死了」。中性技术词，但在 code review 里出现自带警报音。

关　联　活锁、竞态条件、脑裂

活锁

`livelock` · *huó suǒ*

字　面　活的锁——锁着，却还在动。

释　义　各方不断响应对方、改变自身状态，却谁也无法推进，系统「忙碌地原地踏步」；经典比喻是走廊相遇的两人同时向同侧让路、又同时让回来，永远错不开。

例　句　「两边的重试策略互相退避又同时醒来，形成活锁，CPU 打满但一个请求都没成功过。」

> **语言学注**
>
> livelock 是英文从 deadlock 反推造的词，汉译照样仿译，「死／活」这对反义语素严整对仗，构成微型词族——译好一个「死锁」，「活锁」就自动就位，这是术语系统性红利的标本。语义陷阱在于感情色彩倒挂：字面「活」像好消息，实际「活」正是病灶——在动而无进展，比死锁更隐蔽（监控上 CPU 还在跑，进程状态一切正常）。newcomer 需要先破除「动＝健康」的直觉才能读懂这个词。

关　联　死锁、竞态条件、假死、重试风暴

惊群效应

`thundering herd` · *jīng qún xiào yìng*

字　面　受惊的兽群。喻体是草原惊逃（stampede）：一声枪响，整群牛马同时狂奔。

释　义　一个事件（锁释放、连接到达、缓存失效）同时唤醒大批等待者，全体蜂拥争抢，绝大多数徒劳而返，瞬时耗尽资源。

例　句　「早年 Nginx 不开 accept_mutex，一个新连接把所有 worker 全部唤醒，纯惊群，唤而无功白耗调度。」

> **语言学注**
>
> thundering herd 直译应作「雷鸣般的兽群」，中文定为「惊群」是意译中的妙笔：以「惊」点出「同时被唤醒」这个因，比英文状其声势之果更切中机制。后缀「效应」是科学语域包装（同款构式：蝴蝶效应、马太效应），把一幅草原画面提干成术语。newcomer 的难处是喻体只剩「惊」「群」两块碎片，惊逃踩踏的画面感需要补课才能浮现；老手则靠它一词区分「大家都来」与「大家白来」——惊群的痛点不是并发高，而是无效唤醒。

关　联　缓存击穿、重试风暴、缓存雪崩

缓存雪崩

`cache avalanche / cache stampede` · *huǎn cún xuě bēng*

字　面　雪崩——整面山坡的积雪一齐塌下。

释　义　大量缓存条目同时失效（常因设了同一过期时间）或缓存集群整体宕机，请求洪水般直灌后端数据库，将其压垮。

例　句　「所有 key 都设了零点过期，一到点集体失效，缓存雪崩，DB 连接池瞬间打满。」

> **语言学注**
>
> 「雪崩／击穿／穿透」这套三分法基本是中文技术社区（尤其面试文化）自产的分类学，英文并无同样整齐的对应（通行说法是 cache stampede）——罕见的「中文术语体系比源语言更系统」案例。三词共享「缓存＋灾害词」构式，靠灾害意象编码故障模式：雪崩＝面，全体同时失效。四字格在此承担分类功能，一字之差即一种机制，信息压缩率极高，但也把区分的负担全压给记忆——这正是它成为面试黑话身份口令的原因。

关　联　缓存击穿、缓存穿透、雪崩、惊群效应

缓存击穿

`cache breakdown / hot key expiry` · *huǎn cún jī chuān*

字　面　击而穿之。喻体来自电学：绝缘体被高压击穿，电流直通。

释　义　单个热点 key 失效的瞬间，海量并发请求穿过缓存这层「绝缘」直打数据库；防御手段是互斥重建或逻辑过期。

例　句　「明星官宣那条热搜的缓存一过期就是一次缓存击穿，后来给热点 key 加了互斥锁重建。」

> **语言学注**
>
> 借自电学的「介电击穿」，动补紧缩的范本：击（动作）＋穿（结果补语），两个字讲完「高压冲击→防护层失效」的完整叙事，同模板还有打穿、打挂、压垮。在三件套里编码「点」：一个热点、一瞬失效。newcomer 的困难是「击穿」与「穿透」字面近乎同义，术语义却严格互斥，全靠死记「点／无／面」的暗码——近义语素被人为分工，是语义窄化的极端形态。

关　联　缓存穿透、缓存雪崩、惊群效应

缓存穿透

`cache penetration` · *huǎn cún chuān tòu*

字　面　穿透——穿过而无任何阻挡。

释　义　请求查询的数据根本不存在，缓存永远不命中，每一次都穿过缓存直达数据库；常被恶意构造的随机 id 利用。

例　句　「有人拿随机 uid 刷接口，全是缓存穿透，加了布隆过滤器把「查无此人」挡在门外。」

> **语言学注**
>
> 与「击穿」一字之差，分工却严格：击穿是「防护层原来有、被打破了」，穿透是「如入无人之境、从来就拦不住」——前者失效是事件，后者不设防是常态。汉语近义语素（击穿／穿透）被行业征用后强行窄化出互斥义项，词典里查不到这种对立，newcomer 只能靠场景反推。防御语境固定与「布隆过滤器」「缓存空值」「兜底」共现，共现词几乎成了词义的一部分。

关　联　缓存击穿、缓存雪崩、兜底

内存泄漏

`memory leak` · *nèi cún xiè lòu*

字　面　泄漏——液体气体从容器缝隙缓缓漏出。喻体来自水管、油箱。

释　义　程序不断分配内存却不再释放，可用内存持续减少，跑得越久越少，最终 OOM 崩溃。

例　句　「这个服务跑满一周必 OOM，一查是事件监听器没解绑，教科书式内存泄漏。」

> **语言学注**
>
> 仿译 memory leak，流体隐喻精准编码了此类 bug 的时间品格：不是立刻坏，而是慢性失血、难以察觉。细究喻体会发现一个隐藏机关：漏掉的不是被占的内存，而是「可用内存池」里的水位——容器是谁没说破，newcomer 想成「内存漏到别处去了」反而糊涂。正字法上另有一对暗门：资源流失用「泄漏」，信息外泄用「泄露」（数据泄露），一字之差分属可靠性与安全两个领域，混写是外行标记。常与字母词 OOM（Out Of Memory）共现。

关　联　悬垂指针、野指针、僵尸进程

悬垂指针／悬空指针

`dangling pointer` · *xuán chuí zhǐ zhēn*

字　面　悬垂——吊在半空晃荡。绳子一头还攥在手里，另一头拴的东西没了。

释　义　所指对象已被释放，指针变量里还留着旧地址；再解引用即未定义行为，轻则崩溃重则堆被写花。

例　句　「`free` 完没置空，回调里又用了一次，悬垂指针把堆写花了，崩栈完全没法看。」

> **语言学注**
>
> 仿译 dangling pointer。dangle 的意象是「一头系着、一头悬空」，前提是先接受「指针＝拴住对象的绳子」这条根隐喻——根隐喻不装进脑子，「悬垂」就只是个古怪定语，这正是 newcomer 卡壳处。「悬垂」是书面雅词，语言学里 dangling participle 也译「悬垂分词」，同源同调，教科书气十足；口语更常说「悬空指针」，或干脆并进「野指针」不作区分——规范术语与口语在精度上的常态性落差。

关　联　野指针、空指针解引用、内存泄漏

野指针

`wild pointer` · *yě zhǐ zhēn*

字　面　野——野生、无主、不受管束（野狗、野路子）。

释　义　未初始化或已失效、指向不可预知地址的指针；解引用后果随机，是最难复现的一类内存错误。

例　句　「局部变量声明完没初始化就传下去了，纯野指针，每次崩的位置都不一样。」

> **语言学注**
>
> 对译 wild pointer，但汉语「野」的构词能产性远胜英文：野狗、野蘑菇、野路子、野生程序员——「野」作前缀表「无主、未驯化、来路不明」是活模板，所以「野指针」在中文里比 wild pointer 在英文里更常用、更口语（英文教材多半只说 uninitialized/dangling pointer）。感情色彩是贬中带怕，「野」自带失控恐惧。口语常把悬垂指针也并称「野指针」，规范上的二分（未初始化 vs 释放后）被抹平——技术口语中少见的语义扩大案例。

关　联　悬垂指针、空指针解引用、内存泄漏

空指针解引用

`null pointer dereference` · *kōng zhǐ zhēn jiě yǐn yòng*

字　面　解引用——顺着引用去取所指之物。对着「空」取物，自然取不到。

释　义　对值为 `null` 的指针或引用取其内容，触发段错误或 NullPointerException；Tony Hoare 自认的「十亿美元错误」。

例　句　「线上 NPE 告警刷屏，一查是老数据这个字段本来就是 `null`，取值前没判空。」

> **语言学注**
>
> 全链条逐语素仿译的标本：null→空、pointer→指针、dereference→解引用。最硬的骨头是「解引用」：前缀 de-（逆操作）译作「解」，属于数理化语域的构词法（解耦、解卷积），日常汉语几乎不这么用，newcomer 常误读成「解决引用问题」。口语里几乎没人说全称——Java 圈说字母词 NPE，C 圈说「踩空了」，配套的动宾小词族「判空、置空、非空」才是日常战场用语：书面一长串，口头两个字，同一概念的语域双轨。

关　联　野指针、悬垂指针、兜底

兜底

`fallback / safety net` · *dōu dǐ*

字　面　兜——用衣兜、布兜承接；底——最底下。在底下兜着，上面掉下来的东西有人接住。

释　义　主逻辑失败时的最后一道保障：默认值、降级页、备用链路；引申指「出了事最终有人负责」。

例　句　「推荐服务挂了就走热门榜兜底，页面宁可不精准，不能白屏。」

> **语言学注**
>
> 本册少有的非翻译词：它不是译外来术语，而是母语者拿市井现成词（财政兜底、家里兜底）给 fallback 概念重新命名——杂技安全网式的意象比「回退方案」之类译词生动太多，于是全面胜出，堪称本土词反向占领外来概念的样本。动宾紧缩，衍生自由：兜底逻辑、兜底数据、兜得住、兜不住。老手偏爱它，因为一词兼指「位置在最底层」与「责任我来扛」；非大陆背景的 newcomer 反而最难查——词典里没有这个技术义项。

关　联　优雅降级、熔断、幂等、空指针解引用

优雅降级

`graceful degradation` · *yōu yǎ jiàng jí*

字　面　优雅地降低等级。「优雅」状人之仪态；「降级」本是职务军衔的下调。

释　义　系统在部分故障或过载时，有计划地关闭次要功能、降低服务质量，保住核心链路可用；与「雪崩式全挂」相对。

例　句　「大促峰值把评论区和推荐位都关了，只保下单支付，这是预案里的优雅降级，不算事故。」

语言学注

仿译 graceful degradation。英文 graceful 在工程语境里只是「从容、不摔跤」（gracefully shut down），译成「优雅」发生了语域升格：系统砍功能本是狼狈事，术语却披上仪态修辞，自带公关话术潜能——「我们不是坏了，是优雅降级了」的戏谑用法由此而生。newcomer 要点破一层：「优雅」的褒义来自与灾难对比，不砍就全崩，砍得有序即为优雅。前端语境另有配对术语「渐进增强」（progressive enhancement），日常微服务口语则简说「降级」。

关　联　兜底、熔断、限流、雪崩

熔断

`circuit breaker` · *róng duàn*

字　面　熔化而断开。喻体是保险丝：电流过大，熔丝自熔切断电路，牺牲自己保护电器。

释　义　下游失败率或超时率超过阈值时，调用方主动切断请求一段时间、快速失败，防止拖垮自身并给对方喘息；随后半开试探恢复。

例　句　「支付渠道超时率破 50%，熔断器直接打开，十秒后半开放几个请求试探，成了再全开。」

语言学注

意译兼喻体偷换：英文 circuit breaker 本是「断路器」（跳闸可复位），中文却取电工学现成词「熔断」（熔断器＝保险丝，烧毁即一次性）——字面机制与工程实现（closed/open/half-open 三态可复位）有细微失配，newcomer 若较真「熔了怎么还能恢复」反而被字面绊倒。动补紧缩：熔（方式）＋断（结果）。2016 年 A 股熔断机制使它破圈成大众词，属技术—金融双栖术语；「半开」「熔断器打开」这些搭配则暴露它骨子里还是开关。

关　联　限流、优雅降级、雪崩、重试风暴

限流

`rate limiting / throttling` · *xiàn liú*

字　面　限制流量。喻体来自水利：闸门控着水流，超过流量就不放行。

释　义　约束单位时间内的请求量，超出部分排队、降级或直接拒绝（返回 429），保护系统不被洪峰冲垮。

例　句　「开放接口按 appId 限流，单 key 每秒一百次，超了直接 429，别把库打穿。」

> **语言学注**
>
> 动宾紧缩的典型：两字封装 rate limiting 五个音节，且全家族都泡在水利隐喻里——令牌桶、漏桶、洪峰、灌数据，直至四字并列格「削峰填谷」（削峰＋填谷两个动宾骈列，承自电力调度，文言骈俪的现代遗产）。「流」是中文网络词汇的根语素之一：流量、上下游、断流。与近邻「节流」（throttle，本自发动机节气门）有微差：限流偏硬阈值拒绝，节流偏主动放慢，口语常混用。中性、运维日常词。

关　联　熔断、背压、重试风暴、优雅降级

背压

`backpressure` · *bèi yā*

字　面　背＝逆、反向（背流、背光）；压＝压力。逆着流动方向传回来的压力。喻体是流体管道：下游阀门收紧，压力沿管子向上游回传。

释　义　下游消费不过来时，把「减速」信号沿数据流逐级反向传给上游，让生产者放慢，而不是任由中间队列堆到爆。

例　句　「消费端一慢，背压生效，上游发送速率自动压下来了，整条流水线没堆积也没 OOM。」

> **语言学注**
>
> 这是术语的二手旅行：backpressure 本是流体力学与内燃机词汇（排气背压），早有定译「背压」，响应式编程把英文词借去，中文圈再把现成汉译搬来——读者拿到的是「隐喻的隐喻」。字面难点全在「背」：既非背部也非背诵，而是「逆、反」这个在现代口语里已边缘化的义项，透明度极低；大数据圈（Flink 社区）干脆改说「反压」，同源异译并存。老手爱用它，因为一词讲清「反向、逐级、自动传导」三个特征，换任何短语都要一句话。

关　联　限流、重试风暴、缓存雪崩

重试风暴

`retry storm` · *chóng shì fēng bào*

字　面　重试掀起的风暴。风暴：大量事物同时猛烈发生、自我增强。

释　义　故障期间海量客户端按各自策略同时反复重发请求，流量逐层成倍放大，把本已不稳的服务彻底压死。

例　句　「下游一超时，三层服务各自重试三次，一次调用放大成二十七次，重试风暴把网关直接打挂。」

语言学注

仿译 retry storm，「X 风暴」是能产构式（广播风暴、告警风暴、中断风暴），气象隐喻编码「量大、正反馈、不可控」——风暴之可怕不在单个雨点而在自我增强，恰合重试的指数放大。这个词的真正价值是给「好心办坏事」的机制命了名：每一次重试都是善意的自救，叠加起来就是集体谋杀，一词完成一次系统思维教学，事故复盘里点出「重试风暴」四个字即完成归因。贬义。配套解法也成词：退避（backoff）、抖动打散（加随机 jitter）。

关　联　雪崩、惊群效应、熔断、限流

雪崩／级联故障

`avalanche / cascading failure` · *xuě bēng*

字　面　雪崩——山坡积雪失稳，一点崩塌带动全面崩塌；级联——一级连着一级。

释　义　一个组件的故障沿依赖链条被超时、重试、资源耗尽逐级放大，最终拖垮整个系统。

例　句　「核心服务一次长 GC 引发超时，超时引发重试，重试压垮下一环，半小时全站雪崩。」

语言学注

同一概念的双词位并存，分工在语域：书面事故报告用仿译词「级联故障」（学术、可分析、能定位第几级），口头惊呼用隐喻词「雪崩」（整体、失控、画面感）——一场事故两套话语，是行业语域分层的活标本。「崩」是中文故障词汇的核心语素：崩溃、崩盘、崩了，单音节即可独立成句（「崩了！」），堪称最经济的灾难播报。注意歧义收窄：加「缓存」是特指（缓存雪崩），裸用「雪崩」则泛指级联故障。

关　联　缓存雪崩、重试风暴、熔断、单点故障

假阳性／假阴性

`false positive / false negative` · *jiǎ yáng xìng / jiǎ yīn xìng*

字　面　阳性＝检出「有」；阴性＝检出「无」；假＝这个结论是错的。整套来自医学检验单。

释　义　假阳性：没病报有病——没故障却告警、正常邮件进垃圾箱；假阴性：有病没查出——真故障被静默放过。

例　句　「这条告警规则假阳性太高，半夜老空响，大家都免疫了，真出事那次反而没人看。」

> **语言学注**
>
> 医学检验术语整体平移进工程，仿译保留「假＋检验结果」结构。难点是双重否定的心智体操：假阴性＝「说没有」是假的＝其实有——每次都得现场推演，英语使用者同样挣扎，可见难度来自概念结构而非翻译。工程口语里有一对更透明的本土替身：「误报／漏报」，动宾双字对仗、望文即义，日常沟通几乎全用它；但一旦进入精确率、召回率的统计语境，仍须切回假阳／假阴与学术接轨——同义词按语域分工而非竞争。告警疲劳（狼来了效应）是假阳性最著名的工程后果。

关　联　静默失败、毛刺、兜底

毒丸

`poison pill` · *dú wán*

字　面　毒药丸。喻体来自谍战：特工随身的氰化物胶囊；引申为「谁吞下谁出事的东西」。

释　义　投入消息队列的特殊消息，消费者读到即执行约定动作（最常见是退出，用于优雅关闭）；也指反复消费失败、堵死队列的坏消息（poison message）。

例　句　「往队列里塞 N 颗毒丸，每个 worker 领到一颗就自行收工，比广播 `kill` 干净得多。」

> **语言学注**
>
> 仿译 poison pill，军事谍战隐喻（金融界「毒丸计划」是同源的平行借用）。妙在褒贬倒挂：本义歹毒，技术里最常见的用法却是善意的停机握手——毒的只是线程的命，属于程序员死亡幽默词族（`kill`、僵尸、孤儿进程、墓碑）的一员，用死亡词汇消解系统操作的抽象，是行业亚文化的语言表征。newcomer 有双重困惑：为何投毒是正常运维、以及「故意投的毒丸」与「意外出现的毒消息」同词异义，须靠语境分辨故意与事故。

关　联　墓碑、僵尸进程、静默失败

副作用

`side effect` · *fù zuò yòng*

字　面　主作用之外附带的作用。源自药理学：药品说明书上的那一栏。

释　义　函数在返回值之外对外部世界的任何改动——写变量、写文件、发请求。函数式语境是中性技术概念；故障语境指「改了不该改的东西」。

例　句　「这个 getter 里居然发埋点请求，纯副作用，难怪单测跑一遍就上报一次。」

> **语言学注**
>
> 仿译 side effect，且汉语早有药学定译垫底，技术义直接搭现成词的车，落地零成本。语义走向罕见：日常义带贬（药的坏处），进入编程反而中性化、精确化——打个日志也算副作用，只是「须显式管理之事」；newcomer 最大误区恰是带着药盒上的贬义来读它，以为副作用必为坏事。而在故障排查语境（「这函数有隐藏副作用」）它又回归贬义——同一个词在同一行业内维持两套感情色彩，靠语境切换，是语域分层的微观样本。函数式的「纯／不纯」是其配套评价词。

关　联　幂等、静默失败、竞态条件

幂等

`idempotent` · *mì děng*

字　面　幂＝乘方；等＝相等。取幂之后仍与自身相等（x · x = x），数学定义的直读。

释　义　同一操作执行一次与执行多次效果相同；工程上是重试与消息去重的安全前提——「重复打过来也不怕」，最后一道去重兜底。

例　句　「支付回调必须做幂等，网关重发三次也只能扣一次款，拿订单号做唯一键兜底。」

> **语言学注**
>
> 罕见的纯数学术语整体搬运：idempotent（拉丁 idem 同＋potens 幂）经数学汉译「幂等」直接进入工程，没有做任何通俗化加工，结果字面透明度趋近于零——「幂」是现代汉语最生僻的高频技术字之一，newcomer 十有八九先把 mì 读成 mí，再对着两个字发呆。但它因不可替代而存活：找不到第二个双音词能精确表达「重复执行、效果不变」。口语解释性替身是「防重」「去重」，正式接口文档必用「幂等」——一词之内完成雅俗分层：面试考读音，实战考订单号。

关　联　兜底、重试风暴、副作用、墓碑

宕机

`down / downtime` · *dàng jī*

字　面　「宕」的古义是拖延、放荡（跌宕、延宕）；此处通常解为英文 down 的记音字。机＝机器。

释　义　服务器或服务整体停摆、不可用；名词化为「一次宕机」「宕机时间」，是 SLA 里被计较到秒的那个词。

例　句　「监控显示主库宕机四十秒，哨兵自动切了从库，期间写请求全部失败。」

语言学注

音译意译同锅的珍品：一般认为「宕」谐 down 之音（早期写作「当机」「down 机」，台湾至今多写「当机」），大陆书面定于「宕」这个文言字，避开「当」的多义并顺手抬高语域——结果造出一个看似古雅、实为洋泾浜的词，也有人拿「宕」的古义「搁置」附会，两说并存恰说明其字源已在使用中磨灭。它填补了正式文书的空位：报告里不能写「挂了」，必须写「宕机」——与口语词构成严格的语域互补分布。

关　联　挂了／崩了／跪了、单点故障、假死、雪崩

挂了／崩了／跪了

`crashed / went down` · *guà le*

字　面　挂＝口语里讳指死亡（人挂了）；崩＝崩塌；跪＝跪地不起、认输。

释　义　程序或服务停止工作的万能口语。「挂了」最通用；「崩了」偏指进程崩溃退出；「跪了」带认输的自嘲拟人。

例　句　「压测刚到一半网关就跪了，重启也起不来，先回滚再说。」

语言学注

死亡婉辞的技术征用与再直白化：汉语用「挂」讳称人死，程序员把它转指进程之死——对机器无须避讳，婉辞落地变成戏称，隐喻引申兜了一个圈。三词感情色彩递进：崩了（客观陈述）→挂了（随意）→跪了（自嘲拟人，怜其不争）。语域纪律严格：仅限口头与 IM，事故通报必须换写「宕机／不可用／异常退出」——一场事故、两套词汇的双轨制，是本行业语域反差最生动的现场。外国学习者的难点：教科书汉语里「挂」查无此义，且它能产出「挂单机不挂集群」这类高度紧缩的行话句。

关　联　宕机、雪崩、假死

单点故障

`single point of failure (SPOF)` · *dān diǎn gù zhàng*

字　面　单一的一个点出故障（则全局瘫痪）。

释　义　系统中缺乏冗余的组件——它一坏，整个系统随之不可用；架构评审的头号猎物。

例　句　「Redis 还是单实例，妥妥的单点故障，上了哨兵才敢过评审。」

> **语言学注**
>
> 仿译 SPOF，但汉语紧缩吞掉了关键的句法关系：英文 single point of failure 是「失效所系的那一个点」，中文「单点故障」表面像主谓结构（单点发生了故障），实际所指却是结构缺陷本身——多数语境里什么都还没坏，说的是「存在这么一个点」的隐患。口语进一步紧缩成「单点」并直接作谓语：「这里有单点」「把单点消掉」，名词径直用成了病名。newcomer 常误以为它描述一次具体事故，其实它是架构诊断书上的术语，中性偏警示。

关　联　脑裂、雪崩、宕机、兜底

假死

`hang / unresponsive` · *jiǎ sǐ*

字　面　假装死亡或看似死亡而未死。喻体来自动物装死与医学假死状态。

释　义　进程还在、端口能连、心跳还发，但业务请求全部无响应——「活着的死」，比彻底崩溃更难检测。

例　句　「JVM 长 GC 进入假死，进程在、TCP 能通，请求全超时，存活探针一路绿灯，最后人肉重启。」

> **语言学注**
>
> 本土意译的胜利：英文只有寡淡的 hang、unresponsive，中文借动物行为学现成词「假死」注入戏剧性，且语义结构精确——假（表象为活）＋死（实质已废），两字编码「监控指标与真实状态背离」这个复杂命题。与「活锁」互为镜像：活锁是看着在动、实则无进展；假死是看着活着、实则不应答。事故语域高频，带无奈的黑色幽默。newcomer 须先分清存活探测与就绪探测两层，才明白「假」字究竟假在哪一层。

关　联　活锁、心跳、宕机、僵尸进程

僵尸进程

`zombie process` · *jiāng shī jìn chéng*

字　面　僵尸——死而不僵、游走人间的尸体（海地巫毒意象，经港片重塑普及）。

释　义　Unix 中已终止、但父进程尚未 `wait` 回收其退出状态的进程，只在进程表里占一行尸位；泛化指一切「该死未死、占着资源」的残留。

例　句　「父进程忘了 `waitpid`，跑一晚上攒了几千个僵尸进程，PID 都快耗尽了。」

> **语言学注**
>
> 仿译 zombie process，Unix 官方死亡隐喻家族（`kill`、孤儿进程、daemon、zombie）的一员，中文全套照收，这组词的猎奇感本身就是 Unix 文化的身份标记。严格义常被口语磨损：真僵尸已经死透（仅剩进程表一行，杀无可杀），俗用却常把「杀不死的活进程」也叫僵尸——恰好用反。隐喻与机制多点对应是它的教学价值：不害人但占位、成群出现、清理须找父辈（回收机制）——一个词把半页操作系统讲义压缩成画面。

关　联　毒丸、内存泄漏、假死、墓碑

丢包

`packet loss` · *diū bāo*

字　面　丢＝遗失；包＝数据包。包裹在半路丢了。喻体是邮政：把网络想成邮路，数据分组是一件件包裹。

释　义　数据包未能到达对端；以丢包率衡量链路质量，是网络排障的第一个追问。

例　句　「跨机房链路丢包率飙到 5%，TCP 疯狂重传，延迟直接拖高一个量级。」

> **语言学注**
>
> 动宾紧缩的极品：两个字封装「数据分组在传输途中丢失」一整句。前提是 packet→「包」的邮政隐喻已经内化，由此全族皆通：发包、抓包（拦下过路包裹开箱检查，画面绝佳）、拆包、粘包。单音动词「丢」把网络故障说得像日常丢钥匙一样轻巧，语域亲民。newcomer 的坑是一字双源：packet 与 package 同译「包」，「丢包」之包是前者，与安装包、依赖包毫无关系——汉译合并了英文的词形区分，碰撞由读者自行消解。

关　联　抖动、毛刺、心跳

第二辑 · 测试与质量

给代码验明正身的一整套仪式，从「冒烟通过」到「黄金语料」。

冒烟测试

`smoke test` · *mào yān cè shì*

字　面　「冒烟」即物体冒出烟来。喻体出自电子硬件行当：新板子焊好先通电，不冒烟、不烧糊，才配做进一步检测；更早一说来自管道工——往管里灌烟，看哪儿漏烟哪儿漏水。

释　义　对新构建做的最低限度可用性验证：装得上、启动得了、主流程能走通即可，不深究细节，常作为进入全量测试或合入前的准入检查。

例　句　「这个包冒烟都没过，首页一点就闪退，先撤回去修，别浪费 QA 的排期。」

> **语言学注**
>
> 典型仿译（calque）：逐词照搬英文隐喻，却把喻体的现场一并丢了——中文日常里「冒烟」只出现在事故场景（发动机冒烟），新人容易反推成「专门把系统搞冒烟的测试」，误当成压测。更拧巴的是命名逻辑：这类测试以**故障症状**命名（冒烟＝失败），通过时却要说「冒烟通过」，字面自相矛盾。以败象为名、以无败象为过，这层转喻只有知道硬件典故的人才觉得自然——它因此成了一块行话试金石。

关　联　冒烟通过、回归测试、门禁、提测。

冒烟通过

`smoke (test) passed` · *mào yān tōng guò*

字　面　「冒着烟通过了」——字面读来像着了火还带烟闯关，荒诞感十足。实为「冒烟测试通过」的紧缩。

释　义　构建通过了冒烟测试，取得进入下一环节（全量回归、提测、合入）的最低资格。

例　句　「今晚的包冒烟通过，明早 QA 上全量回归，大家别再往这个分支塞新东西了。」

语言学注

一次两级压缩：先是「冒烟测试」删去「测试」，「冒烟」经语义窄化独立成名词，专指这项测试（「先跑个冒烟」「冒烟挂了」）；再与「通过」缩合成四字格，念起来带正式通报的节奏，QA 日报里几乎是公文套语——承文言骈俪的四字韵律给口头行话镀了层官样。妙在字面义与所指义完全相反：冒烟本是失败判据，「冒烟通过」却是喜讯。外人须两步解码（冒烟→冒烟测试→其结果），故它天然筛人：一听就懂的必是圈内人。

关　联　冒烟测试、门禁、提测。

回归测试

`regression test` · *huí guī cè shì*

字　面　「回归」＝回到、归向原处。此处指软件「退回」从前的坏状态：新改动弄坏了旧功能。

释　义　改动后重跑既有测试，确认老功能没被改坏；「回归」也名词化指「被改坏」这件事本身——「这是个回归」＝这毛病是新代码引入的倒退。

例　句　「排查完了，这不是新 bug，是上周重构引入的回归——回归用例居然没拦住，得补一条。」

语言学注

意译自 regression，但感情色彩在跨语域时翻了面：汉语里「回归」几乎总是褒义或庄重义（回归祖国、回归自然、回归初心），统计学「回归分析」也价值中立；唯独在测试语境里它是贬义——倒退、变坏。新人望文生义，很难想到「出回归了」是坏消息。另有一层歧义要点破：回归测试是防回归的测试，不是「会回归的测试」，偏正结构里省略了目的关系。口语还常动宾紧缩为「跑回归」，让「回归」独立承担「整套回归用例」之义。

关　联　冒烟测试、快照测试、用例、脆弱测试。

快照测试

`snapshot test` · *kuài zhào cè shì*

字　面　「快照」即快速拍成的照片，摄影词。计算机行当先借去指磁盘、虚拟机的瞬时存档，测试再借一层：给程序输出拍张「照片」存档。

释　义　把某次运行的输出（组件渲染树、序列化结果、生成文件）存为基准，之后每次运行都与之逐字比对，有差异即失败；确认差异合理后由人工更新快照。

例　句　「你改了按钮文案，二十多个快照全红了——先看 diff 是预期变更还是真回归，是预期的就更新快照再提。」

> **语言学注**
>
> 隐喻引申的二次转手：摄影→存储→测试，每转一次，「快」字的本义（快速成像）就淡一分，如今已是化石语素，没人再问快在哪。真正的洞见在照片这个喻体本身：照片忠实记录**当时有什么**，却不判断**应该有什么**——快照测试的断言不是人写的，而是从程序自己的历史输出里翻拍来的，它检验的是「没有未经解释的变化」，而非「正确」。这门技术的软肋（基准可能一开始就错、动辄误报）全写在喻体里了，堪称隐喻即文档。

关　联　回归测试、黄金语料、断言、脆弱测试。

打桩／桩

`stub / stubbing` · *dǎ zhuāng / zhuāng*

字　面　「桩」是打进地里的柱子，「打桩」是地基工程里实打实的工序：楼未起，桩先下。英文 stub 的本义却是残端——铅笔头、票根、树桩，取「削短的占位物」之象。

释　义　用一段写死返回值的替身代码顶替尚未就绪或不便真调的依赖，让被测代码先跑起来；桩只按预设吐数据，不校验交互，与 mock 相对。

例　句　「支付网关的联调环境还没批下来，你先把它打个桩，写死返回成功，把下单主流程跑通再说。」

> **语言学注**
>
> 这组测试替身词里最成功的归化案例。译者拿「桩」对 stub，字面对上了，喻体却悄悄换了场景：英文是「残端」（真物截短后的剩余），中文是「地基桩」（施工前先打下的临时支撑）——后者反而更贴合实践：依赖未建，先打桩撑住，主体工程照常施工。更关键的是构词红利：「打＋桩」是现成的动宾结构，天然能产——「打个桩」「桩打好了」「给它打上桩」，随汉语语法自由屈折；相形之下「模拟对象」四音节名词根本无法动词化。于是「打桩」成了整个替身家族的**通用动词**，哪怕打的其实是 mock。新人的困惑也在此：先听到的是工地意象，没有任何语素提示「假依赖」。

关　联　桩件、测试替身、模拟对象／mock、挡板。

桩件

stub（作为制品的名词） · *zhuāng jiàn*

字　面　「桩」加后缀「件」——件者，零件、部件之件，把动作凝固成一个可清点的制品。

释　义　名词化的桩：充当测试替身的那段桩代码本身。多见于教材、标准文档等书面场合；口语里通常只说「桩」或「打的桩」。

例　句　「规范要求桩件与真实实现挂在同一接口下，跑单测时靠配置切换，不许在业务代码里写死分支。」

> **语言学注**
>
> 搭了汉语计算机词汇里最能产的类推构词便车：「-件」系（软件、硬件、组件、插件、控件、固件、中间件）自「软件／硬件」那对天才译名起，就是批量制造「人工制品名词」的模具。新人即便不识「桩」的典故，看到「-件」也能猜出是某种部件——派生模板自带半透明度。语域上它明显偏书面：「桩件」有教材腔、国标腔，口语说出来微微发僵。同一个东西，写下来是桩件，说出口是桩，正是书面语与口语在构词层分工的活标本；它与「假件」共用后缀，与「打桩」共用词根，在替身家族里占着「文牍名词」这个生态位。

关　联　打桩／桩、测试替身、假件／fake。

测试替身

test double · *cè shì tì shēn*

字　面　「替身」来自影视行当：武戏危险，找体形相仿者替演员上，即武行替身；「测试替身」＝测试这场戏里的替身演员。

释　义　一切顶替真实依赖的测试用对象的**总称**，涵盖桩（stub）、模拟对象（mock）、假件（fake）、间谍（spy）、哑元（dummy）诸类；术语出自 Meszaros《xUnit Test Patterns》，经 Fowler 的文章流布。

例　句　「面试官追问 mock 和 stub 的区别，我说都属于测试替身，一个管校验交互，一个只管喂数据。」

语言学注

难得一见的**无损仿译**：英文 test double 本就借自影视的 stunt double，而汉语恰有现成的行业对应词「替身」，且拜娱乐新闻所赐（文替、武替、光替）至今鲜活——喻体原封不动搬家，甚至比原文更透明（英文 double 一词多义，「加倍」的干扰始终在）。它在这组词里承担**属名**：桩、mock、假件是种名，各自的翻译策略又恰好三岔——借形再隐喻（桩）、原词不译（mock）、本土合成（假件）。但日常口语几乎不用属名，大家直呼种名或干脆全用 mock 概括——「测试替身」主要活在译著、教材和面试里，语域偏书面，一说出口便透出「读过原典」的身份信号。

关　联　打桩／桩、模拟对象／mock、假件／fake、挡板。

模拟对象／mock

`mock object` · *mó nǐ duì xiàng*

字　面　mock 本义模仿、戏拟（mock exam 即模拟考试）；「模拟＋对象」是规规矩矩的意译，「对象」取面向对象之义。

释　义　预先编入「期望交互」的替身：不光返回数据，还记录并校验被测代码有没有按预期调用它（次数、参数、顺序），测试收尾时统一验证。狭义与桩相对；日常用法里 mock 已泛化为一切测试替身的统称。

例　句　「别真发邮件，把通知服务 mock 掉，最后断言它被调用了一次、参数是这个收件人就行。」

语言学注

一词双轨的标本：书面走意译「模拟对象」，口语几乎全盘语码转换，直接用英文 mock，且嵌得极深——「mock 掉」（接结果补语）、「mock 了一份数据」（带体貌与量词）、「mock 平台」（作定语），英语词根已按汉语语法完全屈折。英文取胜的原因很朴素：单音节动词碾压四音节名词，「你把它模拟对象掉」根本说不出口——同族的「桩」能活，正因它也是单音节。另有一处教科书与街头的断层：文献里 mock 与 stub 泾渭分明（mock 校验交互），民间的「mock 数据」「mock 接口」十有八九其实是桩——语义扩张恰与「语义窄化」反向，泛化到失去区分度。新人先学民间用法再读书，必被这套倒挂搞糊涂。

关　联　打桩／桩、测试替身、假件／fake、挡板。

假件／fake

`fake (object)` · *jiǎ jiàn*

字　面　「假」＋制品后缀「件」，字面即「假零件」，直觉联想是假货、山寨配件。

释　义　拥有**真实可用的简化实现**的替身：内存版数据库、本地目录冒充的对象存储——行为是真的，只是实现取巧、不能上生产。亦译「伪造对象」「假对象」，口语常直接说 fake。

例　句　「仓储层的测试别连真库，写个内存假件，一个字典存取就顶住了，跑起来还快两个量级。」

> **语言学注**
>
> 与「桩」的借形再隐喻、mock 的原词不译不同，「假件」是全透明的本土合成词：谁都能秒懂「假的部件」。可透明是有代价的——「假」在汉语商业语境里强烈贬义（假货、假药、造假），而在 Meszaros 的分类里 fake 恰恰是替身家族中**最「真」**的一员：它真干活、有完整逻辑，只是偷了工减了料（褒义的偷）。最能干的替身摊上了最败坏的名字，感情色彩在翻译中错了位；中文说「我写了个假件」总有点自我举报的味道，这也是它口语不敌英文 fake 的原因之一。三兄弟分工至此齐整：桩喂死数据、mock 验交互、假件真干活。

关　联　测试替身、打桩／桩、模拟对象／mock、桩件。

挡板／挡板服务

本土原创，无固定英文源词（近 `mock server / service virtualization`）　· *dǎng bǎn*

字　面　「挡板」是挡住某物的板子：汽车挡泥板、机箱挡板、炉灶挡板——挡住不该过去的东西，纯粹的防护件意象。

释　义　盛行于银行、电信、政企测试圈的本土说法：架一个模拟服务顶替真实外部系统（银联、征信、短信网关），把测试流量「挡」在环境边界之内并按预设回应答；常说「搭挡板」「走挡板」。

例　句　「测试环境连不了真银联，走挡板，报文按接口文档回个 00 成功码，联调先跑起来。」

> **语言学注**
>
> 替身家族里唯一**没有英文源词**的成员——不是译出来的，是行方测试员自己的隐喻引申。造词动机也与旁支不同：桩强调「撑起来」，mock 强调「装出来」，假件强调「简化的真」，挡板强调的是**边界隔离**——把请求挡回来，别放出去。这正是强监管机构的核心焦虑（测试网严禁外联生产），组织现实直接凝进了词根。于是这组近义词成了社会语言学意义上的方言地图：说挡板的多半在银行电信，说 mock 的在互联网公司，说桩的读过教材或做嵌入式，说测试替身的在写书——听人用哪个词，能猜出他的工位在哪栋楼。

关　联　打桩／桩、模拟对象／mock、测试替身、假件／fake。

夹具／fixture

`test fixture` · *jiā jù*

字　面　「夹具」是机加工车间的老词：把工件夹持在机床固定位置上的装置，保证每一刀都落在同样的地方。

释　义　测试运行前搭好、运行后拆净的已知环境与数据：库里的种子记录、临时目录、拉起的依赖服务，及其 setup／teardown 代码。pytest 等框架又把它泛化为「可注入的测试资源」。

例　句　「这条用例时红时绿，查下来是夹具没清干净，上一条用例残留的数据把断言污染了。」

> **语言学注**
>
> 一次「零成本」的翻译：英文 fixture 本就是从机械车间借进 xUnit 的，而汉语机械业早有定译「夹具」，译者只需沿用旧等值词，隐喻便原厂直达——工件夹稳了每刀才可重复，环境夹稳了每次测试才可重复，机理严丝合缝。吊诡在受众换代：如今的程序员多半没进过车间，喻体在他们那里生而即死，「夹具」沦为硬背的黑箱标签——一个隐喻的存亡不取决于译笔好坏，而取决于读者的人生经验里有没有那间车间。口语中不少人干脆说英文 fixture，书面才写夹具，语码随语域切换。

关　联　用例、测试替身、造数、脆弱测试。

造数／造数据

本土原创，无固定英文源词（近 `test data seeding / preparation`） · *zào shù*

字　面　「造数据」的紧缩：造，制造也（亦不免让人想起造假）；数，数据也。

释　义　为测试预备满足前置条件的数据：批量生成实名账号、特定状态的订单、符合报文规范的流水；大厂常设专门的「造数平台」。

例　句　「这个场景要一个实名过、绑了卡、还挂着未完成订单的用户，先去造数平台造一批，别拿生产数据脱敏凑。」

> **语言学注**
>
> 教科书级的动宾紧缩：双音节动词「造数」从「造数据」里压出来，凝固到能做定语（造数平台、造数工具），完成了从短语到词的全程。「数」在此与「数字」之「数」同形，外人可能误听成算术，圈内却零歧义——紧缩词的可懂度从来靠语境社区兜底。有意思的是修辞姿态：英文同义说法都端着（provision、prepare、seed），中文直接一个「造」字，毫不讳言数据是人造的，坦荡得近乎粗野；「造」与「造假」共享语素的微妙尴尬，被行业默契按下不表。这是少见的中文比英文**更短更狠**、不可译方向逆转的案例。

关　联　夹具／fixture、用例、假件／fake。

覆盖率

`(code) coverage` · *fù gài lǜ*

字　面　「覆盖」是空间与军事意象——火力覆盖、信号覆盖，把一片地域整个罩住；「-率」再把它折成百分比。

释　义　测试运行时代码被执行到的比例，按行、分支、函数等口径统计。它度量「测得多不多」，不度量「测得对不对」。

例　句　「门禁卡增量覆盖率 80%，你这个 PR 才 62%，把那几个异常分支的用例补上吧。」

> **语言学注**
>
> 意译加能产后缀「-率」（通过率、命中率、点击率），一眼即懂，也一眼即坏：「-率」类词天然招管理层喜欢，可下指标、可画曲线，于是覆盖率成了古德哈特定律在 QA 界的头号展品。指标一旦被追逐，隐喻的**超额承诺**就现形：「覆盖」暗示罩住即安全，可一行代码被跑到而无断言，是覆而未测——探照灯扫过不等于看清。行话用衍生词自嘲这层空洞：「刷覆盖率」，一个「刷」字（刷分、刷单、刷题）道尽应付指标的全部犬儒。新人往往先信字面，后信「刷」。

关　联　门禁、用例、断言、单测。

契约测试

`contract test / consumer-driven contracts` · *qì yuē cè shì*

字　面　「契约」是文言色彩的「合同」：契，古义近于刻——刻木为凭、剖券为二、两造各执其半，验合以防抵赖；约，缠束、约定。

释　义　把服务间的接口约定固化为可执行校验：消费方声明自己依赖的请求／响应形状，生成契约文件，提供方在自己的流水线里持续验证未违约——以此替代笨重的联调与端到端环境。

例　句　「下游又被我们改字段坑了一回，补契约测试吧，Pact 文件进仓库，两边流水线都挂上校验。」

> **语言学注**
>
> 意译中的语域挑选术：现代汉语的日常词明明是「合同」，术语却取文言雅词「契约」——「合同测试」听着像在审法务文本，市侩且歧义；「契约」自带庄重与抽象，方能承载「系统之间的信义」这层拟人隐喻（同一选择见于「契约式设计」）。更妙的是词源巧合：「契」的本义就是刻符剖半、各执一片、合之则验——一份契约文件、双方各自持有、两端分别校验，Pact 的机制被几千年前的造字现场精确预言。文言词的复活不是怀旧：雅词的抽象度恰好匹配了架构概念的抽象度。

关　联　端到端／E2E、门禁、断言、快照测试。

黄金语料／黄金样本／金标准／ground truth

`golden corpus / gold standard / ground truth` · *huáng jīn yǔ liào*

字　面　三个喻体三个来路：「黄金／golden」承自工程界的 golden master——压盘复制用的母盘，作绝对基准的那一份；「金标准」借自医学的诊断金标准，再上溯货币金本位——金者，不锈不贬之物；ground truth 来自遥感测绘：卫星影像之外，站在**地面上**实测到的真值。

释　义　经人工核验、被当作绝对正确参照的输入—期望输出集。解析器、OCR、模型的测试以它为准绳，任何偏离都按缺陷论处；语料只增不删，改动须人工签核。

例　句　「新解析器必须在黄金语料上全绿才许合入；要改期望值？先走人工复核签字，语料不是你想改就能改的。」

语言学注

一个概念、三条隐喻通道在中文里汇流，选哪条随亚文化而定：NLP 圈说「黄金语料」（「语料」本是语言学术语——本词典写到这儿颇有照镜之趣），医学统计背景说「金标准」，机器学习圈干脆字母词直用 ground truth 或 GT，语码转换毫不客气。三个喻体说的是同一件事：**不可腐蚀**——黄金不锈，大地不欺。这层隐喻在做规范性工作：叫它黄金，等于在喊「别碰」，数据治理的纪律（只增不删、人签才改）全部内置于词根。新人的陷阱在 truth：这真值不是自然界给的，是人核出来、约定俗成的社会制品——所谓地面实况，也得真有人去地面走一趟。

关　联　快照测试、回归测试、断言、覆盖率。

脆弱测试／不稳定测试／flaky

`flaky test` · *cuì ruò cè shì / flaky*

字　面　flake 本义薄片、碎屑（酥皮、掉漆），美式俚语引申指「靠不住、爱放鸽子的人」；「脆弱」「不稳定」是各取一面的两种意译。

释　义　代码未动、结果时过时不过的测试，败因与被测功能无关——异步竞态、时序、共享状态、外部依赖抖动；是团队对 CI 信任度的头号腐蚀剂。

例　句　「这条用例又 flaky 了，重跑三次过两次，先打上隔离标签挪出门禁，别让它天天误伤别人的 PR。」

语言学注

一桩「意译双双失手、语码转换补位」的案例。flaky 的核心语义是**人品式的间歇失信**——flaky 的朋友会临时放你鸽子——把测试拟人成一个不可靠的同事，贬损里带着无奈的熟稔。两个译名各丢一半：「脆弱」偏「一碰就碎」，反而更贴文献里另一个概念 brittle test（过度耦合实现、一重构就崩），译名一出，两概念在中文里悄然合流；「不稳定」保住了间歇性，却把那口怨气滤得干干净净，成了体检报告语体。于是口语索性不译：「又 flaky 了」——英文形容词直接套进「又……了」的体貌框架。借词的铁律再次应验：**丢义可忍，丢情不可忍**。

关　联　快照测试、端到端／E2E、门禁、回归测试。

端到端／E2E

`end-to-end` · *duān dào duān*

字　面　从一端到另一端。「端」在汉语计算机词汇里是劳模语素：前端、后端、云端、移动端、服务端——有多少端，这个词就横穿多少端。

释　义　贯穿真实系统全链路的测试：从用户入口（UI 或 API）一路打到最深处的依赖（数据库、第三方），验证整条路走得通。昂贵、缓慢、易 flaky，故只宜少量部署在金字塔尖。

例　句　「登录这条 E2E 又超时了，预发环境的短信网关一抖，整条流水线跟着红——这种链路真不该全靠端到端兜底。」

> **语言学注**
>
> 仿译保住了英文的「X-to-X」框式，且译得讲究：peer-to-peer 作「点**对**点」，end-to-end 作「端**到**端」——「对」编码对等关系，「到」编码贯穿行程，一字之差把两种拓扑分得清清楚楚。字母词 E2E 更把英语内部的谐音缩写（2＝to）整体借入，与汉语数字谐音（520＝我爱你）异曲同工，故落户毫无违和。新人的真正障碍是一词多义的跨界：此词源出上世纪八十年代网络体系结构的「端到端原则」，如今「端到端加密」「端到端测试」「端到端模型」各说各话，共通的只剩一个图式——**性质须贯穿全程，中间环节一概不得代劳**。抓住图式，诸义自通。

关　联　契约测试、测试金字塔、脆弱测试、冒烟测试。

单测／单元测试

`unit test` · *dān cè*

字　面　「单元」是 unit 的定译（教材单元、住宅单元），最小可独立看待的一块；「单测」取两个成分的首字缩合。

释　义　隔离验证最小可测单位（函数、类、模块）的测试：快、稳、量大，构成测试金字塔的塔基。口语与工单里几乎只说「单测」。

例　句　「这段逻辑抽成纯函数吧，好写单测，省得 mock 一堆依赖，跑一遍毫秒级。」

语言学注

汉语双音节化压强的标准产物：四音节术语循「取各成分首字」的模板压成双音节（单〔元〕测〔试〕），与北大、家电、内推、联调同一条生产线。韵律要求在此凌驾于语义透明度：「单」独用本义是「孤、奇数」，外人可能误读作「只测一次」，但双音节的顺口保证它在口语里全面击溃全称，缩略形式还反哺书面（「单测覆盖率」堂而皇之进了门禁文案），完成从省事到正名的升格。一个词的存活，节律投的票有时比语义还多。

关　联　覆盖率、测试金字塔、打桩／桩、用例。

测试金字塔

`test pyramid` · *cè shì jīn zì tǎ*

字　面　「金字塔」本身是汉译史上的奇译：埃及尖塔状如汉字「金」，遂以**字形**命名。测试金字塔：塔基大量单测，腰部适量集成与契约，塔尖少许端到端。

释　义　关于测试组合比例的形状学主张：越往上越慢、越贵、越脆，故应「下多上少」。头重脚轻的反模式则叫「冰淇淋筒」或「倒金字塔」。

例　句　「咱们现在是标准的冰淇淋筒，全靠 E2E 撑场面，一轮仨小时还老 flaky——得往下补单测，把塔基砌起来。」

语言学注

双层借喻的套娃：pyramid→金字塔已是一次视觉动机的翻译（以字形拟物形，既非意译也非音译的「象形译」，汉语译名史上屈指可数），测试金字塔再整体骑乘其上。它本质上是**一张图的名字**：全部论证力量都压在几何上——底座宽是稳，尖顶重是危，不必读内文，看剪影即得结论。故其衍生词也全是形状词：倒金字塔、冰淇淋筒、沙漏，方法论论战直接打成了几何图形战。对新人它作为图像全透明，作为处方却不自明：塔并不解释**为什么**该下多上少（那是速度与成本的经济学），隐喻给了记忆钩子，没给推理链。

关　联　单测、端到端／E2E、契约测试、脆弱测试。

门禁

`(quality / merge) gate` · *mén jìn*

字　面　「门禁」原指对门户的禁制，今天最鲜活的所指是写字楼与小区的刷卡闸机：门禁卡、门禁系统。禁，是宫禁、禁地的禁，法度森严。

释　义　代码合入主干前必须通过的自动检查关卡：构建、测试、静态扫描、覆盖率阈值……任何一项不过，合入即被机器拒绝。常说「过门禁」「卡门禁」「门禁红了」。

例　句　「我这个 MR 卡在门禁上了，静态扫描报了两个圈复杂度超标，流水线直接拒收。」

> **语言学注**
>
> 意译的高段位玩法——移植而非直译：英文只是干巴巴一个 gate，中文没有译作「门」或「关卡」，而是征用了物理安防的现成合成词「门禁」，于是借来的不是一扇中世纪城门，而是每个上班族天天刷的闸机。这个喻体自带社会学语义：闸机不认人情、只认卡——恰是 CI 门禁的伦理内核（机器把关，不看资历，吵也没用）。「禁」字再垫上一层法令语体的威严。对圈外人它仍首先意味着自家楼下那道闸，「代码门禁」乍听像机房安保。感情色彩是微妙的敬畏掺怨气：被拦者骂骂咧咧，却极少有人主张拆掉它。

关　联　冒烟测试、覆盖率、质量左移、回归测试。

质量左移／测试左移

`shift-left (testing)` · *zhì liàng zuǒ yí*

字　面　「左移」＝往左挪。前提是把研发流程画成一条自左向右的时间轴（需求→设计→编码→测试→发布），把质量活动往左（早期）挪。

释　义　把测试与质量工作前置：需求评审时就定验收标准、用例先行、静态检查、开发自测，而不是等提测后再围堵；与之对偶的还有把监控延伸到线上的「右移」。

例　句　「与其天天堵在提测后返工，不如质量左移——需求评审 QA 就进场，验收标准当场写成用例。」

> **语言学注**
>
> 一个暗中依赖**书写方向**的仿译：时间轴自左向右流，只是横排左起文字文化的作图习惯；汉语固有的时间隐喻走的是上下与前后（上周／下周、提前／推后），「左」在母语里毫无「早」义。所以「左移」必须借甘特图才解得开——隐喻靠图表而非靠语言存活的罕见标本，母语的自然说法本该是「前置」（「测试前置」也确实并行流通）。选「左移」而非「前置」是一种语域声明：DevOps 布道语体、咨询套件味，工程师私下不免腹诽其为 PPT 词。程序员另有会心一笑：`<<` 正是左移位运算符，「质量 << 1」的双关梗由此而来。

关　联　门禁、单测、用例、提测。

断言

assertion / assert · *duàn yán*

字　面　断，裁断、判定（断案、当机立断）；言，话语。「断言」在通用汉语里指斩钉截铁地下结论，常带「未免武断」的告诫色彩（妄下断言）。

释　义　代码或测试中声明「此处必须为真」的检查点，不满足即失败或中止。一条测试的判定核心就是它的断言；没有断言的测试跑得再绿也形同虚设。

例　句　「这条用例跑了半天什么都没断言，过了也是白过——补一句对返回值的 `assert`，顺手把边界也断一下。」

> **语言学注**
>
> 少见的**译名反超原词**：assert 源于拉丁语「主张、揽为己有」，语感平平；「断言」却是文言底子，「断」携着断案的司法威严——一条 assert 就是一次一锤定音的判决。语义窄化干净利落：通用义「贸然下的定论」（可能错，故显轻率）收窄为「机器逐次核验的真值条件」（必须对，故显严谨），感情色彩随语域从贬翻褒——日常劝人「别急着下断言」，测试里却劝人「多下断言」，同一个词横跨两种美德。老手爱它的凝练（「断一下相等」，动词还能单字挣脱），新人若带着「妄下断言」的语感入行，反要愣一愣：原来这里的断言，越武断越好。

关　联　用例、快照测试、契约测试、覆盖率。

用例

test case（与 **UML** 的 **use case** 同形） · *yòng lì*

字　面　用＋例。「例」是案例、判例、先例的例——一个立得住、可援引的具体实例。

释　义　一条可独立执行、有明确输入与期望结果的测试，也是测试管理的计数单位（写用例、跑用例、用例库、一条 P0 用例）。注意与需求建模的「用例」（use case）同形异义。

例　句　「这次改动影响面不小，QA 评估要补四十多条用例，先把 P0 场景的写完再说。」

语言学注

一场由紧缩引发的同形词相撞。「测试用例」本身就有两种可行切分：〔测试〕〔用例〕，或〔测试用〕〔例〕——「供测试之用的例」，后者的「用」是「军用、家用」的后缀用法。日常缩成「用例」后，恰与 UML use case 的定译撞成同形词，全靠说话人身份消歧：QA 嘴里的用例是 test case，需求分析师嘴里的是 use case；在 QA 占多数的语境里，词义已向前者窄化——语义窄化跟着社区人口结构走。「例」的选择暗藏哲学：判例、先例，一个具体个案代表一类行为，恰合测试的本性——测试是例证而非证明（Dijkstra：测试只能证明缺陷存在，不能证明其不存在）。量词用「条」，把它点数成清单上的一行行，纪律感十足。

关　联　用例爆炸、断言、覆盖率、夹具／fixture。

用例爆炸

`(test) case explosion / combinatorial explosion` · *yòng lì bào zhà*

字　面　「爆炸」搬自「组合爆炸／指数爆炸」——参数、状态、路径一相乘，所需用例数瞬间冲天。

释　义　输入维度的组合使穷举测试在数学上不可行；对策是等价类划分、边界值、pairwise、基于性质的测试等「以策略换穷举」。

例　句　「五个开关俩枚举，全组合好几百条，别硬写了——再写就用例爆炸了，上 pairwise 挑正交组合吧。」

语言学注

「爆炸」族科学隐喻（信息爆炸、人口爆炸、组合爆炸）的行业分身：把指数增长体验为一场**暴烈事故**——不是「很多」，是「多到失控、多到具有破坏性」。妙处在这爆炸无主语、无凶手：没人引爆什么，是组合数学自己把自己炸了，恰合工程师的切肤感受——你没做错任何事，是数学与你为敌。它还是个**修辞武器**：排期会上一句「这会用例爆炸」，就把「我们不打算穷举」重新包装成「数学禁止穷举」，一个词同时完成免责与说服。新人易误读为「用例大面积挂掉」的事故现场——它说的其实是写不完，不是跑不过。

关　联　用例、覆盖率、端到端／E2E、断言。

第三辑 · 架构与设计

房子怎么盖，以及为什么会盖成一座「屎山」——从「边车」到「技术债」。

边车

Sidecar · *biān chē*

字　面　「旁边的车」，喻体是摩托车侧面挂载的载人边斗（俗称挎斗、偏三轮）：主车提供动力，边斗搭载额外乘员，同行同停。

释　义　与主应用同生共死的辅助进程或容器，代主应用承担日志、代理、加解密等横切能力，典型见于 Kubernetes Pod 与服务网格的数据面。

例　句　「日志采集别打进业务镜像，在 Pod 里挂个边车容器就行，主服务一行代码不用改。」

> **语言学注**
>
> 标准的仿译（calque），但喻体本身在汉语里是「二手」的——日常汉语管这东西叫「挎斗」「偏三轮」，很少叫「边车」，于是中文开发者往往先学术语、再反推摩托车意象，隐喻的理解路径与英语读者恰好相反。newcomer 的难点正在此：字面「旁边的车」并不提示「同生命周期、非侵入」这些关键语义，得靠架构图补课。语域中性，云原生圈高频。

关　联　服务网格、微服务、可插拔、熔断

技术债／技术债务

Technical debt · *jì shù zhài*

字　面　技术上欠下的债。喻体来自金融借贷：先借钱（走捷径）解燃眉之急，之后连本带息偿还。

释　义　为赶进度采用次优实现所累积的未来返工成本；不还，则以持续攀升的维护成本形式「付息」。

例　句　「这版先硬编码顶上，技术债我记进 backlog，下个迭代腾出手来还。」

> **语言学注**
>
> Ward Cunningham 1992 年提出的金融隐喻，汉译属仿译，落地却异常顺滑：「欠债还钱」「利滚利」「债台高筑」是汉语现成的认知框架，本金、利息、破产整套蕴涵无损迁移，还派生出「还债」「背债」「债留下一任」等能产搭配——搭配网络之丰富是外来隐喻彻底归化的硬指标。感情色彩中性偏无奈；其修辞功能是给「烂代码」提供一个能写进汇报 PPT 的体面说法，与「屎山」构成一雅一俗的语域配对。

关　联　架构首付、屎山、祖传代码、大泥球

架构首付

`Architectural down payment`（前置的架构投入）　·　*jià gòu shǒu fù*

字　面　为架构付的「首付款」。喻体来自按揭买房：先付一笔大头，余款慢慢供。

释　义　项目早期在抽象、边界、基础设施上有意识的一次性投入，用来压低后续「月供」（维护与扩展成本）；是技术债隐喻的正向配对概念。

例　句　「不求一步到位，但消息这层抽象是架构首付，现在省下这笔，以后连本带利加倍还。」

> **语言学注**
>
> 隐喻引申——在「技术债」打开的金融框架内顺势造词，属隐喻家族的系统性扩展（有债就有首付、利息、破产）。有趣的是它在汉语语境里比在英语里更响亮：「首付」是中国城市生活的头号焦虑词，喻体自带全民情感负载，架构师借它一用，说服力立涨。语域偏架构评审与技术分享的「布道腔」，日常闲聊少用。它是二阶隐喻：newcomer 若没先懂「技术债」，此词完全不可解——理解有前置依赖。

关　联　技术债、单体、微服务

脚手架

`Scaffold / scaffolding`　·　*jiǎo shǒu jià*

字　面　建筑工地围着楼体搭的临时钢管架，供工人踩脚搭手，楼成即拆；「脚手」二字点明功能——给脚和手一个落点。

释　义　自动生成项目骨架或起步代码的工具与产物，如 `vue-cli`、`dotnet new`；让人跳过重复的初始搭建，直接写业务。

例　句　「新服务别手搭了，用公司脚手架一键生成，连 CI 配置和目录规范都带好。」

语言学注

仿译，且是喻体完全本土化的幸运案例——中国城市人人见过裹着绿网的脚手架，意象零成本落地（对比「边车」的水土不服）。但有两处语义漂移值得记录：其一，英语 scaffolding 强调「生成出的临时代码」，汉语用法重心移到了「生成工具」本身（「用脚手架起个项目」），转喻从产物滑向工具；其二，真脚手架竣工要拆，而生成的代码往往终身留在仓库——隐喻承诺的「临时性」基本是假的，newcomer 信了字面就会误判其生命周期。中性偏褒（省事、专业）。

关　联　样板代码、造轮子、可插拔

样板代码／模板代码

`Boilerplate (code)` · *yàng bǎn dài mǎ*

字　面　「样板」即供仿照的定型范本（样板房、样板戏）。英文 boilerplate 原指十九世纪印刷业整块铸好、原样复用的钢版（钢材同锅炉钢板，因而得名），后指合同里成段照抄的套话。

释　义　语言或框架强制要求、却几乎不承载业务信息的重复性代码，如 Java 的 getter/setter、成片的异常声明。

例　句　「Java 写个值对象一屏全是样板代码，换 Kotlin 一行 `data class` 就完了。」

语言学注

意译——铸版印刷的喻体在汉语里无从仿译，译者以「样板」替换，但替换有代价：英语 boilerplate 隐含「无脑套话」的贬义，「样板」在汉语里却中性偏褒（样板＝值得学的范例），致使 newcomer 常把「样板代码」误解为「示范代码」，是本册最典型的一处译介失真。民间另有「八股代码／八股文」的归化说法，借科举程文喻空洞的强制格式，贬义到位，堪称比正式译名更准的意译。语域中性，多带抱怨语气。

关　联　脚手架、胶水代码、造轮子

胶水代码

`Glue code` · *jiāo shuǐ dài mǎ*

字　面　像胶水一样把两个原本对不上的零件粘在一起的代码。喻体来自日常粘合剂。

释　义　不含业务逻辑、只在接口不兼容的组件或系统之间做转换、搬运、适配的代码。

例　句　「这个服务没什么正经业务，全是把老 ERP 和新系统粘起来的胶水代码。」

> **语言学注**
>
> 仿译，且在汉语里先有「胶水语言」（Python 的经典标签）后带火「胶水代码」，构成成对借入。胶水的蕴涵选得极准：非承重（不是梁柱，只是粘缝）、易到处溢（胶水代码倾向蔓延）、拆时连皮带肉。感情色彩微贬——说一段代码是胶水，等于说它必要但不体面。newcomer 上手门槛低，难的是分寸感：老手用它区分「该薄的层」与「该厚的层」——胶水该存在，胶水变厚即是架构警报。

关　联　样板代码、门面、防腐层、接缝

银弹

`Silver bullet` · *yín dàn*

字　面　银铸的子弹。喻体来自欧洲狼人传说：唯有银弹能杀死狼人，引申为「一击制胜的神兵」。

释　义　指望一举解决软件工程全部难题的万灵药；几乎只出现在否定句里，典出 Brooks 1986 年论文《没有银弹》。

例　句　「别指望换个框架就能救活这个项目，软件工程没有银弹。」

> **语言学注**
>
> 仿译的「失重」案例：狼人叙事非汉语文化母题（汉语驱邪靠桃木剑、朱砂、糯米），喻体登陆即失去神话支撑，词义只能靠背 Brooks 典故获得——它是「读书人黑话」，不读《人月神话》基本接不住。更麻烦的是撞词：汉语原有「银弹攻势／银弹外交」（银子＝钱，砸钱开路），newcomer 极易串味成「钞能力」。用法上已高度僵化为否定极性词：「没有银弹」近乎固定引语，肯定式「这就是银弹」只作反讽。语域书面、克制，常用来给过热的技术选型泼冷水。

关　联　反模式、微服务、中台、造轮子

面条代码／意大利面代码

`Spaghetti code` · *miàn tiáo dài mǎ*

字　面　像一碗搅在一起的面条：随便夹起一根，扯出半碗。喻体本是意大利面，汉译换成国民主食「面条」。

释　义　控制流与依赖相互缠绕、无法单独理解或修改任何一段的代码，典出 GOTO 时代对无结构跳转的批评。

例　句　「这段回调套回调的面条代码，我改了一个分支，另外三处跟着崩。」

> **语言学注**
>
> 归化式仿译——舍「意大利面」而取本土主粮，缠绕意象无损且更亲切。值得一记的是源语里本有整个「意面家族」：lasagna code（层层叠叠）、ravioli code（封装成小块），汉语只引进了频率最高的一员，隐喻体系在跨语传播中塌缩为孤词——借词不借系，是术语借入的常态。贬义、口语，但比「屎山」文雅，能上评审会。对 newcomer 几乎零门槛，是本册可懂度最高的隐喻之一。

关　联　大泥球、屎山、耦合、上帝对象

大泥球

`Big ball of mud` · *dà ní qiú*

字　面　一大团烂泥滚成的球，无棱无角无结构。喻体出自 Foote 与 Yoder 1997 年的同名论文，形容「随手糊出来的系统」。

释　义　事实上的「无架构架构」：模块边界溶解，任何部分都能摸到任何部分——也是最常见的系统终态。

例　句　「模块边界早烂穿了，谁都在 import 谁，整个后端就是一个大泥球。」

> **语言学注**
>
> 仿译，学院气重——它在英语里是有论文出处的「命名反模式」，汉语使用者也多在架构讨论与技术写作中引用，口语场合会让位给「屎山」。二者的分工是本册一处漂亮的语义对立：「大泥球」是结构诊断（系统长什么样），「屎山」是情绪宣泄（我对它什么感受），一冷一热，一雅一俗。泥球意象对 newcomer 友好，唯一的坑是以为它只说小项目——论文本意恰恰是：多数大型系统就是滚大了的泥球。

关　联　面条代码、屎山、单体、耦合

屎山（代码）

汉语社区原创黑话（英语近义 `crap code`；日语有平行的クソコード） · *shǐ shān*

字　面　屎堆成的山。粪便喻脏臭不可触碰，山喻经年累月的体量与不可撼动。

释　义　庞大、混乱、无人敢动却仍在线上跑钱的遗留代码库；隐含「重构不如不动」的处世哲学。

例　句　「别重构了，这座屎山能跑就行，你在上面雕花有什么意义？」

语言学注

本册少数非译入词，汉语互联网原创（与日语クソコード平行发生而非互译）。构词是名名偏正：「屎」定性，「山」定量，把「恶心」与「大到只能仰望」压进一个双音节词，韵律上完全汉语化。语域为粗俗黑话，论坛与私聊高频，正式文档绝迹——书面场合自动升格为「历史遗留代码」「技术债」，形成整齐的语域阶梯（屎山→祖传代码→遗留代码→技术债）。感情色彩最微妙：贬中带亲昵，程序员骂自家屎山如骂自家孩子，自嘲与共情兼有；还派生「在屎山上雕花」（于烂系统里抠细节之无谓）、「给屎山添砖加瓦」等二级习语，能产性极强，是汉语开发黑话的标本级词条。

关　联　祖传代码、技术债、大泥球、面条代码

祖传代码

汉语原创；对应英语 **legacy code**（教科书译名「遗留代码」）　·　*zǔ chuán dài mǎ*

字　面　祖上传下来的代码。「祖传」本用于秘方、手艺、老宅——世代相传且来历不可考之物。

释　义　接手自前人（且前人多半已离职）的老代码：没文档、没测试、没人说得清原理，但业务离不开。

例　句　「这段祖传代码没注释没测试，原作者三年前就跑路了，谁动谁背锅。」

语言学注

意译兼隐喻引申的神来之笔：官方译名「遗留代码」平直无味，民间却把 legacy 的「遗产」义嫁接到「祖传秘方」这一本土框架上，语域反差立刻起效——用供奉传家宝的敬语说人人嫌弃的烂摊子，反讽自成。「祖传」还免费附赠三层蕴涵：动不得（祖宗之法）、说不清（秘方不外传）、推不掉（继承由不得你），信息量反超源词，是译语增值的罕见案例。戏谑贬义，口语与技术段子高频；对 newcomer 的唯一风险是当真——以为真有什么值得尊敬的传承。

关　联　屎山、技术债、接缝、绞杀者模式

上帝对象／上帝类

God object / God class　·　*shàng dì duì xiàng*

字　面　像上帝一样全知全能的对象：什么都知道、什么都管、什么都插一脚。

释　义　承担过多职责、聚合过多状态与逻辑的类；系统里人人依赖它，它一改全身痛，是内聚失败的极端形态。

例　句　「`MainWindow` 都八千行了，妥妥的上帝类，从渲染到导出什么都归它管。」

语言学注

仿译，宗教隐喻取「全知全能」义，贬义反讽——神性在此是罪名。更有趣的考据在译词自身：「上帝」本是先秦典籍里的至上神名，明清传教士借它译基督教的 God（后有著名的「译名之争」），四百年后这枚旧译又被程序员拿去给反模式命名——一词身上叠着两次借用的地层。newcomer 的坑是褒贬误判：字面听着威风，实为代码评审里的死刑判词。口语书面通用，评审场景高频。

关　联　内聚、耦合、单体、充血模型

贫血模型

Anemic domain model · *pín xuè mó xíng*

字　面　「贫血」是医学病名：血量不足，面色苍白，有形无力。喻指只有躯壳、没有生命力的对象模型。

释　义　领域对象只装数据（一堆 getter/setter）而无行为、业务逻辑全被抽去放进 service 的建模风格；Fowler 视之为反模式，实践中却是主流。

例　句　「咱们 domain 层全是 getter/setter，业务全堆在 service 里，标准的贫血模型。」

语言学注

仿译，医学隐喻：对象该有的「血」（行为）流失了。汉译的精彩处在配对——源语是 anemic 对 rich（贫血对「富」，隐喻框架断裂），汉语却造出「贫血／充血」的严整医学对偶，靠汉语构词对对称的偏好，把源语没有的系统性补齐了，可称译语的范式压强修复了源语缺口。小瑕疵：医学上「充血」其实也是病（如结膜充血），细究则两头皆病态，但没人细究。语域是 DDD 圈行话、论坛论战的常年火药桶；newcomer 的难点在价值判断——它是反模式还是务实默认，业内至今无共识。

关　联　充血模型、六边形架构、上帝对象

充血模型

Rich domain model · *chōng xuè mó xíng*

字　面　血量充盈的模型。与「贫血」相对：气血足，自己能动。

释　义　数据与行为同居于领域对象、业务规则内聚在实体与值对象里的建模风格，DDD 的正统立场。

例　句　「订单的状态流转写进实体里，走充血模型，别什么都甩给 `OrderService`。」

语言学注

严格说是汉语的逆构词：英语只有 rich domain model，「充血」是为对仗「贫血」而生造的医学词形，源语并无对应说法——这是汉语术语系统「宁造词也要成对」的典型标本，对偶偏好升格为构词动力。代价是字面歧义：医学「充血」是病理（红肿），此处却取「充盈」的褒义，学过生理学的 newcomer 反而容易被专业知识绊倒。语域同「贫血模型」，二词成对出现，几乎不单说。

关　联　贫血模型、六边形架构、洋葱架构

六边形架构／端口与适配器架构

`Hexagonal architecture (ports and adapters)` · *liù biān xíng jià gòu*

字　面　六条边的多边形。名字来自 Cockburn 画架构图时随手选的形状——多画几条边好摆放多个端口，「六」并无深意。

释　义　核心域居中、一切外设（数据库、UI、消息）经由端口与适配器接入的架构风格，正名「端口与适配器架构」。

例　句　「按六边形架构来：HTTP 和数据库都是适配器，核心域一行框架代码都不许 import。」

语言学注

「以图命名」（视觉转喻）的代表：术语指称的是白板上那张图而非思想本身，「六」这个偶然细节被词形永久固化——中文课堂上「为什么必须六个边」是保留提问，仿译忠实地把源语的误导性一并搬来。正名「端口与适配器」语义透明却干瘪，传播上完败给怪名字：术语竞争里，可记忆性常压倒可解释性。语域书面、架构讨论；对 newcomer 属于「名字越具体越迷惑」的典型。

关　联　洋葱架构、充血模型、防腐层、接缝

洋葱架构

`Onion architecture` · *yáng cōng jià gòu*

字　面　像洋葱一样层层同心包裹的架构。取洋葱的同心层结构：依赖只能由外层指向内层。

释　义　Palermo 2008 年提出的分层风格：领域模型居核，外圈是应用服务与基础设施，与六边形架构、整洁架构大同小异。

例　句　「新服务按洋葱架构分层，依赖一律向内指，domain 摆在最核心。」

语言学注

仿译，又一例「以图命名」——六边形、洋葱、整洁架构三张图讲的近乎同一件事，术语却各立山头，newcomer 以为是三种技术，实为三次命名。汉语这边另有词源彩蛋：「洋葱」的「洋」正是近代汉语给舶来品打的标签（洋火、洋楼），一个自带借入史的词又来翻译一个借入的架构，层层包裹，恰如其名。口语里还有源语没开发的笑点：「剥洋葱架构，剥一层哭一层」——催泪的蕴涵是汉语用户自行引申的。中性书面语。

关　联　六边形架构、充血模型、防腐层

单体／巨石应用

`Monolith / monolithic` · *dān tǐ*

字　面　单一整体。英文 monolith 源自希腊语「独块巨石」；汉译丢了石头，只剩「单个身体」。

释　义　整个应用构建、部署为一个进程或制品的形态；与微服务相对——本是软件的默认形态，微服务时代才被追认为一种「架构风格」。

例　句　「团队就五个人，老老实实写单体，别一上来就拆微服务。」

语言学注

意译，且撞上在位科学术语——化学里「单体」是 monomer（聚合成高分子的最小单元），与「铁板一块」的本义几乎相反，跨专业的 newcomer 常在此打结。另有保留巨石意象的竞争译法「巨石应用／巨石架构」，生动但小众，再次印证高频抽象译名淘汰低频形象译名的规律。感情色彩随时代摆动：微服务狂热期是贬义（「还在写单体？」），退潮后被「雄伟的单体」平反，如今说「先写单体」反而透着务实老练——一个词的褒贬史就是一部架构风尚史。

关　联　微服务、分布式单体、大泥球、绞杀者模式

微服务

`Microservices` · *wēi fú wù*

字　面　微小的服务。micro- 对译「微」。

释　义　把系统拆成一组可独立开发、部署、伸缩的小服务，围绕业务能力划界，服务间经网络通信。

例　句　「先把用户和订单拆成两个微服务，其余模块等边界看清楚了再动。」

语言学注

仿译，其汉语落地有一个漂亮的时机巧合：术语走红的 2014 年前后恰逢汉语「微」前缀的能产爆发期（微信、微博、微商、微电影），「微服务」空降进一个现成且时髦的构词槽位，毫无外来感——借词成活率取决于目标语是否恰好留着形态学空位，此为教科书案例。注意「微」只是相对单体而言，多小算微并无标准，newcomer 追问粒度是必经之路。感情色彩坐过山车：从银弹级热词到「拆过头」的集体反思；动词搭配「拆」（拆服务、拆过细）已沉淀为固定用法。

关　联　单体、分布式单体、服务网格、中台

分布式单体

`Distributed monolith` · *fēn bù shì dān tǐ*

字　面　「分布式的单体」——自相矛盾的组合：身体拆开了，神经还连在一起。

释　义　形式上拆成多个服务、实质仍强耦合的系统：一次需求波及多个服务、必须一起发版；兼得微服务的运维成本与单体的耦合，两头吃亏。

例　句　「服务拆了十几个，可一个需求得同时改八个仓库一起上线，这就是个分布式单体。」

语言学注

矛盾修饰法（oxymoron）术语化的标本：贬义全靠语义冲突自身发力，无须再加贬字。它是诊断词而非风格名——但 newcomer 极易误读成一种可选架构：字面看不出反讽，得先知道「分布式」「微服务」「单体」在行业话语里各自的褒贬站位，才听得出弦外之音。这种「必须懂行才听得出的贬义」是黑话排他性的温和形态。架构评审高频，杀伤力大：一句「这是分布式单体」可以否掉整套拆分方案。

关　联　微服务、单体、耦合、服务网格

服务网格

`Service mesh` · *fú wù wǎng gé*

字　面　服务织成的网。mesh 本指网眼、网状织物——彼此交织、没有中心的连接形态。

释　义　把服务间通信（路由、重试、熔断、加密、观测）下沉到基础设施层的方案：每个服务旁一个边车代理组成数据面，加统一控制面。

例　句　「上了服务网格，重试和熔断从业务代码挪进边车，SDK 总算能瘦身了。」

> **语言学注**
>
> 仿译带轻微意象走形：英语 mesh 是不规则网眼（mesh network 译「网状网络」），汉语「网格」却偏向横平竖直的格子（网格纸、有限元网格），翻译把乱网规整化了。更微妙的是语境撞车——「网格」在当代汉语公共话语里高频指向「网格化管理」，两个「网格」一个管服务一个管人，倒都干着「统一管控」的活，无意间构成跨域回声。云原生书面语，口语常直接说 mesh、Istio（语码转换）；newcomer 需先懂边车与控制面／数据面这组配套仿译，理解链条较长。

关　　联　边车、微服务、熔断、分布式单体

中台

汉语原创；英文媒体只能回译作 **middle platform / middle office** · *zhōng tái*

字　　面　前台与后台之间的「中台」。空间三分：前台接客，后台管家，中台居中供能。

释　　义　把多条业务线共用的能力（用户、交易、数据等）沉淀为可复用的共享层；源自阿里 2015 年「大中台、小前台」战略，后随「拆中台」风潮由圣典跌为教训。

例　　句　「业务还没跑通就先建中台，什么能力都往里沉淀，纯属本末倒置。」

> **语言学注**
>
> 本册罕见的汉语原创架构术语，且实现了反向输出——英文世界只能笨拙地回译成 middle platform 再加注解释。构词是类推填空：既有「前台／后台」这对现成方位词，范式里「中」的槽位天然空着，一填即成，毫无新词感——这正是它传播神速的形态学原因。语用上它是「大厂黑话」的旗舰：与「赋能」「沉淀」「抓手」结伴出没，被无数黑话讽刺文点名，感情色彩从万众朝圣到群嘲反噬，堪称行业对术语通胀的一次清算。newcomer 的难点不在字面而在所指飘忽：中台到底是架构、组织还是 PPT 概念，业内至今各执一词。

关　　联　微服务、单体、银弹

门面／外观（Facade）

Facade pattern（法语 **façade**，建筑正立面） · *mén miàn*

字　　面　铺子临街的脸面。汉语「门面」本指店面房；习语「撑门面」＝维持体面外观。

释　　义　GoF 结构型模式：给一组复杂子系统包一个简单统一的入口，调用方只见门面、不见后厨。

例　　句　「给这堆老接口包一层门面，新代码只准调门面，谁也别再直接摸底下的 SOAP。」

> **语言学注**
>
> 归化意译的上品：façade 的本义（建筑立面、体面外观）与汉语「门面」几乎逐义对齐，连「只顾表面」的微妙贬义都同构，习语「撑门面」白送了整个模式意图——译入语固有词让术语增值，可遇不可求。且存在双译并立：GoF 中译本作「外观模式」（书面正统），口语与 Laravel 等框架圈偏爱「门面」，一词书口两栖，是观察译名竞争的活样本。newcomer 的坑在把门面当万能中转层——门面该薄，厚了就长成上帝对象。

关　联　防腐层、胶水代码、接缝、上帝对象

接缝

`Seam` · *jiē fèng*

字　面　两块布、两块板相接的缝。缝纫喻体：缝线可拆可改，不必剪坏布料本身。

释　义　Feathers《修改代码的艺术》中的概念：不改动某处代码本身、就能改变其行为的切入点（如可注入的依赖），是给遗留代码补测试的下刀处。

例　句　「先在这个静态调用处开条接缝，把时钟注入进去，才能给这段祖传逻辑补上测试。」

> **语言学注**
>
> 仿译，喻体日常（衣缝、瓷砖缝）流通却极窄——它是「书籍绑定词」：几乎只在读过那本书（或其中译本）的人群里通行，可见术语存活靠社群接力而非翻译质量，本条是反面教材式的证据。语义上它把抽象的「可测试性」空间化为「可拆开的缝」，比 testability 可操作得多，老手极爱；newcomer 没读过原典则完全听不出术语义，只当是在聊装修。中性书面语，测试与重构语境专用。

关　联　可插拔、耦合、祖传代码、门面

防腐层

`Anti-corruption layer（DDD，缩写 ACL）` · *fáng fǔ céng*

字　面　防止腐蚀、腐烂渗入的隔离层。汉语「防腐」日常见于防腐剂、管道防腐涂层。

释　义　DDD 战术模式：在自己的领域模型与混乱的外部或遗留系统之间加一层翻译，把对方的概念转成己方语言，防止外来模型污染核心域。

例　句　「对接老 ERP 时加一层防腐层，把它那套字段翻译成我们自己的领域语言再放进来。」

语言学注

仿译中的取义抉择：英语 corruption 一词双关——政治腐败与物质腐蚀，汉译落在「腐蚀／腐坏」一侧，喻体变成工程防腐涂层，丢了官场讽喻，却换来更贴切的画面（隔绝渗漏、保鲜），是译名比源词更聚焦的例子。理解门槛在「腐」的抽象化：被防的不是烂代码本身，而是对方概念体系的渗透——newcomer 常把防腐层等同于适配器，其实它防的是语义污染，比门面、适配器多一层建模野心。DDD 圈书面语，与「限界上下文」等配套出现。

关　联　门面、接缝、六边形架构、绞杀者模式

绞杀者模式／绞杀榕模式

`Strangler fig pattern` · *jiǎo shā zhě mó shì*

字　面　「绞杀者」即绞杀榕——热带榕属植物，种子在宿主树上发芽，气根层层包裹宿主，经年累月宿主枯死，榕树成形。

释　义　渐进替换遗留系统的策略：新系统贴着老系统生长，流量一块块切给新实现，直到旧系统被完全「绞杀」下线，避免推倒重写。

例　句　「老单体别推倒重写，按绞杀者模式来，新服务一块一块把流量接过去。」

语言学注

仿译加截短造成的隐喻变形：Fowler 原词是 strangler fig（绞杀榕），取其「缓慢、自然、新旧共存」的植物学温柔，他本人后来还特意补上 fig 以淡化暴力感；汉语通行形却掐掉了「榕」，「绞杀者」听着像连环杀手，喻体从雨林生态滑向凶案现场，温和的演化叙事在词形上失传。这提醒我们：截短是有语义代价的（对比「造轮子」截掉「重复」反而增值）。全译「绞杀榕模式」存在但小众。书面语，系统迁移方案专用；newcomer 若不识这种植物，只能记结论、丢意象。

关　联　单体、微服务、防腐层、祖传代码

反模式

`Anti-pattern` · *fǎn mó shì*

字　面　「反的模式」。anti- 对译「反」，承自设计模式（design patterns）运动的命名传统。

释　义　一种被反复采用、看似合理、实则有害的常见做法，并被起了名字以便识别与规避——大泥球、上帝对象皆属此列。

例　句　「把配置塞进单例全局乱读，这是典型反模式，往后测试都没法写。」

语言学注

仿译，借了汉语能产的「反~」前缀（反例、反证），但语义常被 newcomer 误析：反模式不是「不按模式办事」，而是「反面教材本身也成了模式」——它照样是复现的、有名字的 pattern，只是价值为负；「反」修饰的是好坏而非有无。这一命名传统体现了模式运动的核心信念：给坏东西起名字才能讨论它、才能在评审里一词毙之——命名即治理。中性偏学术，评审与技术写作高频；判定某做法「是反模式」时常带盖章定罪的语用力量。

关　联　大泥球、上帝对象、银弹、分布式单体

黄金路径／快乐路径

Golden path / happy path · *huáng jīn lù jìng*

字　面　金子铺的路／一路顺心的路。两个英文源词：happy path 出自测试圈（一切顺利的那条执行流）；golden path（又称 paved road，铺好的路）出自平台工程（官方铺设、全程护航的推荐用法）。

释　义　不出任何差错的理想执行流程；引申指平台方修好的「官道」。与之相对的是错误分支与边界情况那些「野路」。

例　句　「Demo 只跑通了黄金路径，超时、重复提交这些分支全没处理，别急着上线。」

语言学注

汉语把 happy path 与 golden path 两条来路合流成近义词对，源语中「测试术语」与「平台工程术语」的分工在译入后趋于模糊——一词多源的归并是术语翻译常见的熵增。「快乐路径」是拟人仿译（路本身无所谓快乐，快乐的是一路无坑的执行和写它的人）；「黄金」则接上汉语「黄金周、黄金时段」的贵重联想。语用要点：这组词多用于揭短——「只测了快乐路径」是委婉的验收批评，newcomer 听字面以为是表扬，实为警报。

关　联　关键路径、脚手架、反模式

关键路径

Critical path · *guān jiàn lù jìng*

字　面　决定成败的那条路径。「关键」本义是门闩与机括（关＝门闩，键＝插销），皆是牵一发而动全身的小部件。

释　义　源自运筹学关键路径法（CPM）：任务依赖网络中最长的那条链，决定整体工期；软件语境引申指请求链路或渲染流程中决定延迟的主干（如关键渲染路径）。

例　句　「登录接口在首屏关键路径上，多加一次远程调用都得掂量掂量。」

> **语言学注**
>
> 本册的「老资格」：它经二十世纪工程管理与运筹学文献进入汉语，早于互联网术语潮几十年——提醒我们技术汉语是分层堆积的，并非都是网络时代的仿译新贵。词义在软件圈发生语义窄化与转移：从项目排期（时间依赖）滑向性能链路（延迟构成）。与「黄金路径」构成一对必须辨析的假姊妹：黄金路径论「顺不顺」，关键路径论「等多久」，同缀「路径」而语义正交，newcomer 极易混为一谈。书面中性，性能优化与排期讨论两栖。

关　联　黄金路径、耦合、熔断

造轮子／重复造轮子

Reinvent the wheel · *zào lún zi*

字　面　轮子早已发明，你又造一遍。喻体是人类最古老的机械发明——轮。

释　义　全称「重复造轮子」指弃现成方案不用、自行重写（贬义）；截短的「造轮子」在中文社区已翻转为中性甚至自豪——亲手实现一个库以练功或补缺；「轮子」更独立成名词，指自研组件。

例　句　「业务代码里别重复造轮子，直接用现成库；真想练手，周末自己造个轮子玩。」

> **语言学注**
>
> 本册语义演变最完整的标本：仿译进入（reinvent the wheel，贬）→截短（掐掉「重复」）→词义改良（造轮子＝练手造库，可写进简历）→名词化外溢（「我写了个轮子」「轮子哥」）。截短并非简单省字：删去「重复」二字恰好删掉了原习语的全部贬义负载，剩下的「造轮子」重新成为一门手艺——英语说 I reinvented a wheel 无论如何不体面，汉语「我造了个轮子」却可以是炫技。动宾结构入句灵活（造过轮子、轮子造多了），口语戏谑；褒贬取决于是否带「重复」，newcomer 须按词形辨极性。

关　联　银弹、脚手架、样板代码

内聚

Cohesion · *nèi jù*

字　面　向内聚拢。物理学早有「内聚力」（分子间相互吸引）的译名，软件沿用其字。

释　义　一个模块内部各成分彼此相关、共同服务于单一职责的程度；结构化设计（1970 年代）中与耦合并提的度量，越高越好。

例　句　「把校验逻辑挪回订单模块吧，讲究点内聚，别让工具类什么都管。」

> **语言学注**
>
> 承袭自然科学在先译名的意译，软件义属语义移植而非新造。它在汉语技术教育里的真正存在形式，是六字口诀「高内聚低耦合」——承文言骈俪之风的反义对偶，念着上口，像乘法口诀一样被背进面试与述职，也像口诀一样常被脱离操作含义地复诵：能背者众，能指着代码说清「这里为什么不内聚」者寡。口诀化是汉语教学传统赋予外来术语的独特命运：记忆性拉满，思考性易归零——newcomer 之难不在懂词，而在破咒。

关　联　耦合、正交、上帝对象

耦合

Coupling · *ǒu hé*

字　面　「耦」本义二人并肩共耕（《论语》「长沮、桀溺耦而耕」），耒字旁泄露了农具出身；「耦合」即成对相联。

释　义　模块间相互依赖的紧密程度：一处变更迫使另一处跟着变，谓之耦合高；架构工作的半壁江山就是降它。

例　句　「支付和通知耦合得太死，中间加个事件总线解耦一下。」

> **语言学注**
>
> 罕用字的技术再就业：「耦」在现代汉语里几乎只活在这一个词中（先经物理学译 coupling——电磁耦合、耦合振子——再入软件），不少程序员为写对这个字专门学它，堪称被单一术语供养的字；常见误写「偶合」实为另一词（巧合义）。真正能产的是反义动词「解耦」：动宾紧缩、及物好用，近年更溢出技术圈进入大众话语（组织解耦、生活解耦）——术语出圈的典型路径：先专业、后隐喻、终至鸡汤。书口两栖，词性中性，但评审语境里「耦合」几乎总作贬用。

关　联　内聚、正交、接缝、分布式单体

正交

Orthogonal / orthogonality · *zhèng jiāo*

字　面　正相交——成直角。线性代数译名：两向量内积为零，互不含对方的分量。

释　义　两个设计维度互不影响：改动其一，无需惊动其二。《程序员修炼之道》将其立为设计总纲，是耦合概念的数学化升格。

例　句　「权限和缓存这两块设计上是正交的，谁改谁都不用惊动对方。」

> **语言学注**
>
> 数学隐喻的声望借用：说「解耦」是工程腔，说「正交」自带线性代数的学历光环——语义近似而语域分层，术语选择成了身份展演。理解它需要一次真正的概念跳跃：从「垂直」到「无关」之间隔着「投影为零」的几何直觉，没修过线代的 newcomer 只能死记「正交约等于互不影响」，隐喻对他们是纯密码；而日常滥用又使它语义漂白，渐成「不相干」的花哨同义词。书面语，架构与 API 设计语境，褒义（正交是设计美德）。

关　联　耦合、内聚、可插拔

抽象泄漏／漏水的抽象

Leaky abstraction · *chōu xiàng xiè lòu*

字　面　抽象层像管道一样渗漏——本该密封在下层的实现细节，从缝里渗了上来。

释　义　Spolsky 定律：所有非平凡的抽象都有漏。ORM 之下的 SQL、TCP 之下的丢包，总有一天你得掀开抽象去看底下那层。

例　句　「ORM 用得再顺，一碰到 N+1 查询你就得亲自读 SQL——抽象泄漏了。」

> **语言学注**
>
> 仿译，水暖隐喻。汉语落地时暗合一个正字法细节：「泄漏」（流体外渗）与「泄露」（机密外传）是近音分工的异形近义词，此处取「漏」，与「内存泄漏」同框（而「数据泄露」取另一边）——可抽象漏出来的偏偏是下层的「秘密」，两个写法竟都说得通，用字选择悄悄站了物理隐喻这边。传播上它借了「内存泄漏」的先入之势，词形似曾相识，接受成本低。对 newcomer 的价值是祛魅：封装不是结界，用抽象者终须懂其下一层——本册少数能直接改变学习策略的词。书面、克制，带一点无可奈何的宿命感。

关　联　门面、防腐层、正交

可插拔／插件化

Pluggable / plugin architecture · *kě chā bá*

字　面　可以插上、也可以拔下。喻体来自电气接插件与热插拔（hot swap）硬件。

释　义　把某类能力做成按统一接口接入的组件，增删实现无需改动核心；「插件化」指整个系统按此原则组织的做法。

例　句　「导出格式做成可插拔的，以后新加一种模板不用动核心流程。」

> **语言学注**
>
> 汉语构词法的技术展示柜：「插拔」是反义联合式复合动词（同类有开关、呼吸、买卖），两个相反动作并置表「可逆的接入能力」——比英语 pluggable 只说「可插」多编码了「可拔」半层意思，一次难得的译语增益；「可~」前缀对译 -able，「~化」后缀对译 -ization，加上名词「插件」（插＋件，「件」是组件、控件、插件一族的能产语素），一个词族三种构词法同台。语义上它是接缝、正交等抽象原则的落地形态，对 newcomer 反而最友好：USB 的体感人人都有。中性偏褒，方案设计高频。

关　联　接缝、正交、脚手架、边车

第四辑 · 流程与协作

代码离开你的手、走向主干的一路关卡，从「提 PR」到「甩锅」。

提交

`commit` · *tí jiāo*

字　面　「提出并交付」——公文语感，向上递交材料。

释　义　把暂存的变更固化为版本历史上一个带说明的节点；名词与动词同形（「一个提交」「提交代码」）。

例　句　「这个提交拆成三个吧：重构归重构，修 bug 归修 bug，格式化单独一个，评审的人好看。」

> **语言学注**
>
> 意译，但英文 commit 的核心义「作出承诺、落子无悔」（commit to sth.）在「提交」里荡然无存，中文选了公文意象，庄重却丢了心理重量。更大的坑是同形异义：日常「提交表单」是发送即达，Git 的提交只是本地存档、离推送还差一步——「我提交了你怎么看不到」是新手第一课。名动同形令量词搭配别扭（「一个提交」），口语因此大量语码转换说「一个 commit」「commit 拆一下」。

关　联　合入、压平、回退、拉取请求（PR）／合并请求（MR）

合入／合并

`merge` · *hé rù / hé bìng*

字　面　「合」是汇合，「入」是进入——支流汇入干流的水利意象。

释　义　把一个分支的提交并进另一个分支（通常是主干）；「合入」偏指「进入目标分支」这一结果，故常带宾语：「合入主干」。

例　句　「这个 MR 两个 approve 都有了，赶在封版前合入 main，不然又得等下个发布窗口。」

> **语言学注**
>
> 「合并」是 merge 的标准意译，对称而中性；「合入」是行话里的动补紧缩变体，补语「入」加进方向与结果义，天然指向目标分支，暗合「汇入干流」的隐喻，两个泛义语素在此凝固窄化为版本控制专义。初学者的困惑在两形并存而分工微妙：文档写「合并」，评审对话说「合入」「先别合」；单字缩略「能合了吗」是口语经济性的极致——谁说「合入」，谁多半天天在提 PR。

关　联　主干、拉取请求（PR）／合并请求（MR）、冲突、压平

冲突／合并冲突

`merge conflict` · *chōng tū*

字　面　两股力量迎面相撞；日常汉语里是人际、武装级别的大词。

释　义　两条分支改了同一处代码，工具无法自动合并，须人工裁决取舍；解决动作叫「解冲突」。

例　句　「你先别合，咱俩这两个 PR 改了同一个文件，按老规矩谁后合谁解冲突。」

> **语言学注**
>
> 意译兼语义窄化的样本：从「武装冲突」级别的大词窄化为「同一行被两边改了」这一极具体的机器判定。词的暴力语感放大了事件严重度，新人初见满屏大写 `CONFLICT` 常有一瞬真实的惊恐，其实多数冲突三分钟能解。动词搭配「解冲突」的「解」承自「解题、解结」，暗示这是智力活而非事故，恰好把语感拉回中性。「谁后合谁解冲突」是挂在这个词下面的一条不成文分布式礼仪。

关　联　合入、变基、主干、拉取请求（PR）／合并请求（MR）

变基

`rebase` · *biàn jī*

字　面　「变」改变、「基」基底——把一串提交搬到新地基上重建，建筑意象。

释　义　Git 操作：把当前分支的提交依次重放到另一个基点之上，得到一条没有合并节点的直线历史。

例　句　「你这分支落后 main 二十多个提交了，先 rebase 一下再提 PR，别拖一串 merge commit 进来。」

> **语言学注**
>
> 教科书级仿译（calque）：base 对「基」、re- 对「变」，逐语素对应，且「基」延续了建筑隐喻——换了地基，整栋楼原样重砌。难点是双向的：不懂 Git 的人从「变基」二字读不出任何操作；懂 Git 的人又嫌它书面拗口，于是口语几乎总是语码转换——「rebase 一下」「公共分支别 rebase」，「变基」多存活于译著与中文文档。这种「书面仿译词＋口语原词」的分层并存，是本册最典型的社会语言学景观。

关　联　主干、拣选、压平、重置

拣选／樱桃挑选

`cherry-pick` · *jiǎn xuǎn / yīng táo tiāo xuǎn*

字　面　英文原义「只摘熟樱桃」——果园里专挑好果子，喻「只挑对自己有利的」。

释　义　把其他分支上的个别提交单独复制到当前分支，而不合并整条分支；常用于往发布分支挑补丁。

例　句　「主干上修空指针那个提交，单独 cherry-pick 到 release-1.3 分支，今晚的热修复要带上它。」

> **语言学注**
>
> 英文本是带贬义的习语（辩论中「只挑有利证据」），入 Git 后已中性化；中文「拣选」走文雅意译路线，「拣」「选」同义并列，自带文言与《圣经》和合本的神学语感——一个诙谐的果园俚语被译成了庄重的书面词，语域反差鲜明。直译「樱桃挑选」几乎只见于译文。此词口语语码转换率大概是全册最高：动词直接说「cherry-pick 过去」，因为「拣选一下」听着像布道。初学者从「拣选」二字也确实猜不出「复制单个提交」这层操作义。

关　联　变基、热修复、合入、主干

压平／拍平／squash 合并

`squash merge` · *yā píng / pāi píng*

字　面　squash 本义「压扁、挤扁」——把一摞东西压成薄薄一张，手上功夫的意象。

释　义　合并分支时把多个提交压缩成一个再合入，使主干历史一条记录对应一个完整功能。

例　句　「你这 PR 里十几个 `fix typo` 的小提交，合的时候记得 squash 掉，别污染主干历史。」

> **语言学注**
>
> 「压平」「拍平」是罕见的拟态意译：不造庄重术语，直接用手部动作对译英文的身体隐喻，「拍」字尤其口语（像拍扁一个易拉罐），保留了原词的粗粝幽默。此词至今无权威定译（压缩合并、压平、拍平并存），而译名竞争未决恰是语码转换的温床——口语几乎总说 squash。它还编码了一种历史观：主干历史是给人读的叙事而非流水账；懂了这层价值观，才懂有人为何对「一个 PR 一个提交」有洁癖。

关　联　合入、变基、拉取请求（PR）／合并请求（MR）、主干

主干

`trunk / main`（旧称 `master`）　·　*zhǔ gàn*

字　面　树的主干。版本历史像一棵树：主干笔直向上，分支旁逸斜出。

释　义　仓库的主开发线，今多名为 `main`（SVN 时代叫 `trunk`，Git 旧默认叫 `master`）；一切分支从它出、最终回它去。

例　句　「我们走主干开发，特性分支活不过三天，没做完的功能靠开关兜底。」

> **语言学注**
>
> 一次完整的植物隐喻移植：trunk 对「主干」、branch 对「分支」，中文恰有现成的同构词族，仿译毫不费力，属于翻译最幸运的情形。词汇层里沉积着年代地层：同一对象可叫「trunk／master／main／主干」，选词暴露说话人的入行年代与技术栈出身。更妙的是，英文圈因 master 的主奴联想而改名 main，中文「主干」隔岸观火——植物隐喻没有那层历史包袱，译词躲过了原词的政治风波。「主干开发」已凝固为四字工程术语。

关　联　特性分支、合入、变基、封版

特性分支

`feature branch`　·　*tè xìng fēn zhī*

字　面　为某一「特性」（功能）岔出去的一条「分支」——树杈意象的延续。

释　义　为开发单个功能从主干切出的短命分支，做完经评审合回主干即删。

例　句　「每个需求单独开特性分支，统一 `feature/` 前缀命名，合完就删，别攒一堆僵尸分支。」

语言学注

仿译词，但译点有瑕疵：feature 在产品语境本该译「功能」，译者选了更学究的「特性」，初学者遂常问「特性分支和功能分支是两种东西吗」（不是）。口语常整个绕开：「开个分支」「切个 feature」——量词「个」直接支配英文名词，是句内语码转换的标本。动词「切」（从 checkout／switch 引申）与「分支」的搭配是行话的隐形门槛：圈外人听「切个分支」，如闻黑话。

关　联　主干、拉取请求（PR）／合并请求（MR）、合入、上线

拉取请求（PR）／合并请求（MR）

`pull request / merge request` · *lā qǔ qǐng qiú / hé bìng qǐng qiú*

字　面　「请求（维护者）拉取（我的分支）」——视角是分布式的：我无权推进你的仓库，只能请你来拉。

释　义　把分支变更提交评审、待批准后合入目标分支的那个工单与页面；GitHub 叫 PR，GitLab 叫 MR，实为一物。

例　句　「我下班前把 PR 提了，你明早帮我看一眼，凑够两个 approve 才让合。」

语言学注

全译名「拉取请求」是逐词仿译，但日常口语几乎百分之百用字母词：「提个 PR」「PR 挂了三天没人看」——动词「提」（「提交」的紧缩）支配字母词宾语，是汉英混合动宾结构的教科书样本。初学者有双重困惑：一是方向反直觉，明明想把代码「推」给你，术语却说「请你拉」，须懂分布式权限模型才通；二是 PR 与 MR 名异实同，跨平台跳槽常被同事纠正。「PR」在中文职场还兼指公关，消歧全靠行业语境。

关　联　代码评审／代码审查（CR）、合入、特性分支、压平

代码评审／代码审查（CR）

`code review` · *dài mǎ píng shěn*

字　面　「评」评议、「审」审查——像审稿一样审代码。

释　义　合入前由同事阅读变更、提意见并批准的流程；也指这件事本身（「等 CR」「过 CR」）。

例　句　「这个改动逻辑有点绕，CR 的时候麻烦重点看第三个 commit 的边界条件。」

> **语言学注**
>
> 「评审」与「审查」是一对意译竞争形：前者借自学术圈的同行评审（peer review），色彩平等协作；后者带行政检查的威权语感——不少团队刻意选「评审」，以示审的是代码不是人，选词即立场。口语大量缩为字母词并动词化：「帮我 CR 一下」。初学者的障碍不在词义在语用：评审意见里 nit、可选、必须改各有潜台词，读不出分级，就会把客气当命令、把命令当客气。

关　联　拉取请求（PR）／合并请求（MR）、合入、自测、无责复盘

回退／还原

`revert` · *huí tuì / huán yuán*

字　面　「回」返回、「退」后退——往回走一步。

释　义　用一次新的反向提交抵消某个已有提交的改动；历史不删不改，只是「以进为退」。与「重置」相对：回退留痕，重置抹历史。

例　句　「那个提交把线上打出 500 了，先 revert 掉止血，根因留到明天复盘再挖。」

> **语言学注**
>
> 意译，但中文字面动作（后退）与 Git 实际动作（前进一步、内容反向）恰好相反——术语隐喻与实现机制脱节的标本。初学者总以为 revert 会「删掉那个提交」，直到看见历史里多出一个 `Revert "..."` 才醒悟。「往回」义近词三分天下：「回退（revert）」在代码历史层做反向提交，「重置（reset）」直接改写历史，「回滚（rollback）」在部署层切回旧版本——三词各守一层，混用是新手暴露身份的高频信号。

关　联　重置（reset --hard）、回滚、热修复、复盘

重置（reset --hard）

`git reset --hard` · *chóng zhì*

字　面　「重」重新、「置」放置——把指针重新放到某处。

释　义　把分支指针强行移到指定提交，`--hard` 连工作区一并抹平，其后的提交从历史上「消失」。是改写历史的操作，与「回退」的加法哲学相反。

例　句　「本地这几个提交全废了，直接 `reset --hard origin/main` 拉平重来，别一个个 revert 了。」

> **语言学注**
>
> 「重置」本是电器说明书用语（重置路由器），语感无害；移入 Git 后按下的却不是复位键，而是没有确认框的时光机——词的语感安全度与操作的实际危险度严重错配，酿成无数新手血案。老手因此发展出仪式性的完整念法「reset --hard」，参数从不省略，像念危险品全名。与「回退」的分野必须点透：回退是「多走一步来抵消」，重置是「假装没走过」——公共分支上只许前者。

关　联　回退／还原、回滚、变基、主干

回滚

`rollback` · *huí gǔn*

字　面　「回」向回、「滚」滚动——车轮倒转、卷轴倒卷的机械意象。

释　义　把线上系统整体退回上一个已知良好的版本或状态（发布回滚、事务回滚）；作用层级在部署与运行态，与改代码历史的「回退」「重置」分工不同。

例　句　「新版本错误率飙了，先别查了，立刻回滚到 v2.3.1，保命要紧。」

> **语言学注**
>
> 仿译（roll 对「滚」、back 对「回」），「滚」保留了原词的滚动意象，动感十足；其日常粗俗义（滚出去）在术语语境里完全休眠，是语境屏蔽本义联想的好例子。语用上这是个应急动词：事故处理的第一反应是回滚而非查因（先止血、再验伤），它在告警群里出现时总伴随肾上腺素。与「回退」的层级之别（动生产环境还是动版本库）是新人必修的辨析。

关　联　回退／还原、上线、灰度发布、热修复

流水线

`pipeline` · *liú shuǐ xiàn*

字　面　工厂流水线——工件在传送带上依次经过各道工序。

释　义　代码从提交到构建、测试、打包、部署的自动化工序链；一次执行也叫「跑一条流水线」。

例　句　「你刚推的提交把流水线跑挂了，单测红了六个，先修绿再想合入的事。」

语言学注

英文 pipeline 本是输油管道隐喻，汉译却换成工厂流水线——不是仿译而是喻体替换式意译：目标语挑了本文化更熟的意象（福特流水线家喻户晓，「管线」反而隔膜；台湾多译「管线」，保留了原管道喻）。这一换极成功，还附赠动词搭配：「跑流水线」「流水线挂了／红了／绿了」，把自动化执行拟作机器开动、把结果染上信号灯颜色。初学者易把它与「持续集成」混为一谈：流水线是机制载体（那条传送带），CI 是实践理念（勤进厂检验）。

关　联　持续集成（CI）、持续交付／持续部署（CD）、提测、上线

持续集成（CI）

`continuous integration` · *chí xù jí chéng*

字　面　「持续地集成」——不断把各人的代码汇到一起验证。

释　义　每次提交都合入主干并自动构建、跑测试，让集成问题尽早暴露的实践；口语中常直接代指那套自动化系统。

例　句　「CI 又排队了，二十分钟还没轮到我这条，构建机真该扩容了。」

语言学注

双音步四字格意译，工整但高度抽象——「集成」在日常汉语几乎不单用（多数人只在「集成电路」里见过），初学者从字面只读出「一直在组装什么」。更有趣的是转喻漂移：术语本指一种实践，日用中却几乎总指那台系统（「CI 挂了」「等 CI 绿」），理念词被工具义架空——问「你们做不做 CI」与「CI 红没红」的，已是两个语义层。「红／绿」借测试报告颜色转喻成败，是伴生的微型隐喻系统。口语几乎只说字母词。

关　联　流水线、持续交付／持续部署（CD）、合入、自测

持续交付／持续部署（CD）

`continuous delivery / continuous deployment` · *chí xù jiāo fù / chí xù bù shǔ*

字　面　「交付」是把货送到客户手上；「部署」是军语，把兵力摆上阵地。

释　义　持续交付指每个版本随时可发布（发不发由人拍板）；持续部署更进一步，流水线自动发到生产环境。缩写同为 CD，常被合并含糊使用。

例　句　「我们号称 CI/CD，其实 CD 只自动到预发环境，真上生产还得人工点那个按钮。」

语言学注

一个缩写罩住两个术语，是字母词歧义的典型案例——英文社区自己都缠斗多年，中文「CI/CD」连读成套话后，多数使用者已不再区分，也常常不必区分。「部署」的军事隐喻值得玩味：deploy 从「展开兵力」引申到「把软件摆上服务器」，中文恰好用同一个军语对译，仿译严丝合缝。初学者的难点在于：交付与部署的差别是「谁按发布键」这一责任边界，藏在词义缝里，字面读不出来。

关　联　持续集成（CI）、流水线、上线、灰度发布

上线

`go live / release to production` · *shàng xiàn*

字　面　登「线」——网络线路、生产线与「上前线」三重意象的共振；与舞台「上场」同构。

释　义　把版本发布到生产环境、真实用户可用的动作与时刻；也作名词：「今晚有个上线」。

例　句　「周五不上线是铁律，真要破例，就得留人盯监控盯到后半夜。」

语言学注

动宾紧缩（上＋线），中文互联网业最核心的仪式动词。「线」的多义共振托住了它的庄重感：上线是有风险的出征，因此衍生出整个短语族——上线窗口、上线评审、灰度上线、紧急下线（「下线」为其反义对偶）。英文得看语境在 deploy、release、launch、go live 之间挑词，中文一个「上线」全包，是难得的「中文一词覆盖英文数词」的方向逆转。初学者易混「发版」与「上线」：包发出去可以不放量，上线才意味着与用户见面。

关　联　灰度发布、封版、热修复、回滚

热修复

`hotfix` · *rè xiū fù*

字　面　「热」即带电、运行中——机器不停机，趁「热」修理；源自电工的「带电作业」。

释　义　绕过常规排期，对线上紧急缺陷立即修复并发布的小版本；也指承载它的分支（`hotfix/`）。

例　句　「支付回调丢单了，拉个 hotfix 分支就改那两行，今晚必须上。」

语言学注

仿译，「热」直接继承英文 hot 的「运行中、带电」义，同族有热更新、热重载、热插拔，自成一个「热-」词族。这个「热」是初学者的假朋友：日常汉语的「热」联想到温度或流行（热搜），读不出「不停机」义，必须整族习得。语用上「热修复」自带紧急、加班、绿色通道的语义韵——一说「出 hotfix」，全员神经收紧，感情浓度远超字面。

关　联　打补丁、上线、回滚、拣选

打补丁

`patch` · *dǎ bǔ ding*

字　面　衣服破了缝块布上去——汉语固有的缝纫意象。

释　义　对已发布的软件追加小修改包；也泛化指对任何东西做临时小修补（给系统打补丁、给流程打补丁）。

例　句　「这版先打个补丁顶着，彻底的重构排到下个迭代再说。」

语言学注

英文 patch 本义就是补丁，中文直接激活自家现成的缝纫隐喻，连动词都是现成的——「打」沿用「打个结、打铁掌」的手艺义，动宾浑然天成，是外来概念彻底本土化的范例（没人会说「安装一个补丁」以外再造新词）。感情色彩微妙：中性之下有一丝将就、临时的贬义暗流，「补丁摞补丁」直接成了形容技术债的生动骂语。对初学者全无障碍，对老手则是表达无奈时最顺手的词。

关　联　热修复、上线、回滚

封版／代码冻结

`code freeze` · *fēng bǎn / dài mǎ dòng jié*

字　面　「封」查封封存、「版」版本——把这一版封起来，不许再动；freeze 则是「冻住」的冰雪意象。

释　义　发布前的截止节点：此后只收阻断级修复、不收新功能，直到版本发出。动作与状态同词：「周四封版」「封版期间」。

例　句　「明晚八点封版，没合入的需求自动顺延下个版本，别再找我开口子。」

> **语言学注**
>
> 「封版」是印刷出版业旧词的转世——报纸付印前版面「封版」，同族有「截稿」；这是旧行业词整体平移而非仿译的少见案例，比直译「代码冻结」资历老、汉味足，口语也更偏爱它。「冻结」则自带一套冰的隐喻（冻住、解冻）。它是流程词里权力色彩最重的一个：封版即关门，之后合码要走例外审批，「赶在封版前」因此成为每个版本周期末集体冲刺的语言引信。

关　联　上线、提测、合入、里程碑

里程碑

`milestone` · *lǐ chéng bēi*

字　面　古时官道旁记里程的石碑——走到哪儿了，一望便知。

释　义　项目计划中标志阶段性完成的节点或版本目标；在 issue 系统里也是一个把任务归堆的容器。

例　句　「把这批 issue 都挂到 1.0 里程碑下面，月底前清零。」

> **语言学注**
>
> 仿译（mile 对「里程」、stone 对「碑」），且是渗入通用汉语最深的一个：政治报道、婚礼致辞都说「里程碑意义」。工程语境里它反而经历了语义窄化兼去崇高化——从「伟大节点」降格为项目管理软件里可增删的下拉选项（「建个里程碑」「里程碑延期了」）。庄重词被日常化的落差常让新人隐隐违和：碑，怎么还能延期呢。

关　联　排期、冲刺、封版、积压

排期

`scheduling` · *pái qī*

字　面　「排」排列、「期」日期——把日子排出先后；承自影视院线的「排期」「档期」。

释　义　给需求、测试、上线安排时间档位的动作与结果；「排期满了」即近期无档。

例　句　「技术评审是过了，但测试排期排到下下周，产品那边你去对齐一下预期。」

> **语言学注**
>
> 汉语原生的动宾紧缩词，从影剧行业平移而来，把工程时间重构成档期经济：人力是场地、需求是节目，插队叫「加塞」，让路叫「顺延」。英文没有对等的日常单词（scheduling 太泛、slot 太碎），这是中文流程词汇比英文更细密的一例。它也是权力话语的缓冲垫：「排期紧张」常是拒绝的礼貌式——听懂潜台词（＝这周别想）是职场语用的必修课。

关　联　里程碑、冲刺、提测、封版

冲刺

sprint · *chōng cì*

字　面　短跑最后阶段的全速奔跑——田径隐喻。

释　义　Scrum 中固定长度（常为两周）的开发周期，以评审和复盘收尾；国内团队常与「迭代」混称。

例　句　「这个冲刺容量满了，你这需求进下个 sprint，先搁 backlog 里排着。」

> **语言学注**
>
> 仿译自田径术语，但隐喻先天带着悖论：短跑冲刺靠不可持续的爆发，而 Scrum 恰恰标榜可持续节奏——英文社区自嘲「没人能连续冲刺」，中文全盘继承了这个缺陷，且「冲刺」在管理层口中常悄悄滑回字面义（＝加班冲一把），成为敏捷话语被异化的语言证据。口语里 sprint、迭代、冲刺三形并用：「迭代」最中性，选哪个词往往暴露团队的敏捷血统是 Scrum 正统还是土法自创。

关　联　站会、积压、复盘、排期

积压／待办

backlog · *jī yā / dài bàn*

字　面　「积」堆积、「压」压着——积压的订单与公文，汉语固有的仓储文牍意象。

释　义　产品全部待做事项的有序清单；条目叫「积压项」，或干脆说「backlog 里的东西」。

例　句　「这个想法不错，先记进 backlog 排优先级，别插队进本迭代。」

语言学注

翻译上的老大难：英文 backlog 在 Scrum 里是中性词（就是一份清单），中文「积压」却继承了原词浓重的贬义语义韵——积压意味着失职、该清未清；「待办」又太轻，像张便利贴。两个译名各偏一头谁也没赢，结果是此词口语语码转换率居高不下：「放 backlog」「理一下 backlog」。初学者望「积压」生义，以为是坏事要清零；其实健康的产品积压恰恰永远清不完——词的感情色彩与概念的价值判断当面打架。

关　联　冲刺、看板、站会、排期

看板

Kanban（经日语「看板」回借） · *kàn bǎn*

字　面　日语「看板」本义招牌、告示牌——丰田工厂里传递补货信号的卡片。

释　义　可视化任务流的方法及那块板子本身：任务卡片在「待办—进行中—完成」诸列间流动，并限制在制品数量。

例　句　「站会别念流水账了，直接开着看板过：评审列卡着的这三张，今天谁能捞走？」

语言学注

本册唯一的日语借形词：「看」「板」本是汉语语素，东渡日本组合成「招牌」义，再随丰田生产方式与敏捷运动回流中文——一次词汇的环球旅行，而英文只能音译 Kanban。中文读者望文生义恰好能通（看的板子），借得毫无阻力，以致多数使用者根本不知道它是外来词。语义也从「补货信号卡」漂移为「任务可视化板」，制造业原义只剩考据价值。「过看板」「捞卡片」等动词搭配自成一套微型行话。

关　联　积压、站会、冲刺、对齐

站会／每日站会

daily stand-up · *zhàn huì*

字　面　站着开的会——站姿本身就是议程约束：站累了，会也该散了。

释　义　敏捷团队每日一次的短同步会：各自讲昨天做了什么、今天做什么、有什么阻塞。

例　句　「站会就三句话：昨天、今天、卡点，别当场展开讨论，散会再拉人单聊。」

语言学注

对 stand-up meeting 的仿译兼紧缩，妙在「站」＋「会」两个语素拼出的新词完全自释——用身体姿势给会议限时，喻体一目了然。但能指与所指早已脱钩：远程时代人人坐在工位上开「站会」，词照用不误，这是理据消亡而词形存续的「语言化石化」现场标本。全称「每日站会」在口语中紧缩为双音节「站会」，顺应汉语词汇双音化的强大引力。

关　联　冲刺、对齐、复盘、看板

无责复盘

`blameless postmortem` · *wú zé fù pán*

字　面　「无责」＝不追究个人责任地「复盘」。

释　义　源自 Google SRE 文化的事故回顾原则：默认当事人在当时信息下做了合理决策，只追系统与流程缺陷、不点名问罪，以此换取真话。

例　句　「说好了无责复盘，大家把当时的判断如实讲——我们要修的是流程，不是修理谁。」

语言学注

混合构词：blameless 意译为「无责」（法律文书语感的文言缩合），postmortem 则直接套用已本土化的「复盘」——英文的验尸隐喻在中文里被彻底洗白成棋局隐喻，译词竟比原词更契合「无责」的温和立意，算翻译的意外增值。语用风险在言行落差：这个词本质是一句承诺，若会上照旧追责，它立刻沦为反讽素材（「无责复盘，责在盘后」）。它与「甩锅」构成镜像词对：一个从制度上消解追责，一个在人性里转嫁追责。

关　联　复盘、甩锅、背锅侠

联调

`joint debugging` · *lián tiáo*

字　面　「联合调试」的紧缩——两方以上把各自的模块接起来一起调。

释　义　前后端、上下游服务或跨团队系统在接口层对接并联合排错的阶段，通常排在各自自测之后、提测之前。

例　句　「接口文档先冻结，下周三开始联调，到时候谁的字段对不上文档谁改。」

> **语言学注**
>
> 典型的双音节紧缩造词（联合调试→联调），与「提测」「自测」同族。英文并没有一个地位相当的日常词（integration 太重，joint debugging 没人这么说），可见这是汉语原生的流程阶段命名，中文开发词汇自成体系的证据。「调」读 tiáo 不读 diào，是入行的隐形语音门槛，念错即暴露新人身份。语用上「联调」自带扯皮预期——接口两侧互指对方不符文档，所以现实对话总是先小人后君子。

关　联　提测、自测、对齐、甩锅

提测

`submit for QA testing` · *tí cè*

字　面　「提交测试」的紧缩——把版本正式提给测试团队。

释　义　开发完成且自测通过后，移交测试人员进入测试阶段的动作与节点；常伴随提测邮件、提测单。

例　句　「今天下班前必须提测，不然测试排期赶不上周四的封版。」

> **语言学注**
>
> 动宾紧缩造出的双音节动词（提交＋测试→提测），是中文流程词的标准构词模板，同族有联调、上线、封版。它还编码着一个组织事实：开发与测试是两个有正式交接仪式的角色——在开发自测自发的团队里，这个词根本不存在，因此它是观察团队分工形态的语言化石。初学者常搞错方向：「提测」是开发提给测试，不是测试提出问题；而且它是有承诺效力的里程碑动词，说出口就咬合进整条排期。

关　联　自测、联调、封版、排期

自测

`self-testing` · *zì cè*

字　面　自己测自己（写的东西）。

释　义　开发者在提测或提 PR 前对自己改动做的验证，从跑单测到手点主流程；「自测通过」是移交下游的门槛承诺。

例　句　「PR 描述里把自测点列一下：正常流、超时重试、权限不足这几个分支都点过了吗？」

语言学注

与「提测」同模的紧缩词，但语用重心在责任修辞：「我自测过了」等于签下一张个人信用支票，出了低级 bug，这句话会被原样引用来问责——于是催生出仪式化书面形态：PR 模板里的「自测清单」。词义的微妙在边界游移：自测到什么程度才配提测，各团队标准悬殊，社群默会知识不写在词里；新人把「编译通过」当「自测通过」而挨训，训的其实是对同一个词的不同践约标准。

关　联　提测、代码评审／代码审查（CR）、联调

对齐／拉齐

`align / sync up` · *duì qí / lā qí*

字　面　排版与队列用语——让参差不齐的东西站到同一条线上。

释　义　互联网职场话术：就目标、认知、进度或口径达成一致；「拉齐」更强调把落后一方主动拽到同一水位。

例　句　「这个方案先跟产品对齐一下预期，别开发到一半需求又变了。」

语言学注

align 的仿译，叠加剧烈的隐喻引申与语义窄化：先从排版的物理对齐引申到抽象认知域，再在职场语域里窄化定格为「达成一致」，宾语无所不包（对齐目标、对齐口径、对齐颗粒度）。「拉齐」是动补变体，那个「拉」泄露了权力语义——有人拉、有人被拉。它是「互联网黑话」批判浪潮的头号靶词：语义稀释到近乎虚词（一切开会皆可曰对齐），却又真实填补了「把各方理解摆到同一水平面」的表达空缺，故禁而不绝。初学者的困难不在懂而在信：第一次听「咱们对齐一下」，总疑心有什么精密动作，其实就是聊聊。

关　联　站会、联调、复盘、排期

甩锅

`pass the buck / blame-shifting` · *shuǎi guō*

字　面　把背上的锅甩给别人——「锅」是「黑锅」的紧缩，背黑锅即代人受过，甩锅即把黑锅扔出去。

释　义　出问题时把责任推给别人（人、团队或上游依赖）；开发语境常见于事故定责与联调扯皮。

例　句　「日志都在这儿，接口是他们改的又没同步文档，这锅我们不背——不是甩锅，是事实。」

语言学注

市井熟语「背黑锅」的现代衍生链：先把「黑锅」紧缩为单音节「锅」，再配上动感极强的「甩」，一个抛物线动作把抽象责任具象成实体道具；随后整个「锅」词族爆炸式派生——接锅、送锅、锅从天降、这口锅太大。语域是纯口语、戏谑兼贬义：正式复盘文档绝不会出现，但复盘会前后的走廊对话里密度极高。它与「无责复盘」构成制度与人性的对照词：后者存在的意义就是让前者失业。对外国同事这是最难解释的词之一，得先补「背黑锅」的民间典故底色。

关　联　背锅侠、无责复盘、联调、复盘

背锅侠

scapegoat · *bēi guō xiá*

字　面　「背锅」加「侠」缀——背起黑锅的「侠客」，仿「蜘蛛侠」「钢铁侠」的译名构词。

释　义　总被迫或主动替团队承担责任的人；也用于自嘲：「我就是全组的背锅侠。」

例　句　「每次出事故都拉他去顶，妥妥的背锅侠，年终奖倒是一分没多。」

语言学注

三重构词叠加：黑锅→锅（紧缩），背锅（动宾），再以类词缀「-侠」派生成名词。「侠」缀本由超级英雄译名带火，褒义崇高；粘到「背锅」上便产生强烈语域反差——用英雄尊号册封受气包，悲壮又滑稽，反讽全靠这个错位。感情色彩随人称移动：自称是苦中作乐，称人则在同情与讥讽之间摇摆。对译词 scapegoat（替罪羊）同样是隐喻：羊是祭祀意象，锅是市井厨房意象——两种文化各自挑选的「代人受过」道具，是绝佳的跨文化比较素材。

关　联　甩锅、无责复盘、复盘

第五辑 · 性能与容量

与流量赛跑的词汇，大多泡在一套「把请求当水」的治水隐喻里。

吞吐／吞吐量

throughput · *tūn tǔ ／ tūn tǔ liàng*

字　面　吞进、吐出。现成的港口航运词——「港口吞吐量」指码头一年装卸多少万吨货。

释　义　系统单位时间能处理的请求量或数据量，衡量总处理能力，与「延迟」构成性能的两根轴。

例　句　「单实例吞吐也就撑到两千 QPS，再往上加压，延迟就开始飙了。」

> **语言学注**
>
> 译者没有仿译 through＋put，而是整体挪用航运经济里现成的「吞吐量」，属意译中的旧词新用。「吞吐」本身是汉语典型的反义联合式复合词（同「呼吸」「收支」「出入」），一进一出两个反义动词并置，天然编码「双向总流量」——这层构词便利是英语 throughput 没有的。喻体把系统拟作一张嘴、一座港，生动但含混：初学者最大的困惑是单位不明——吞吐的是请求数、字节数还是行数？词形完全不给提示，全靠上下文。

关　联　QPS／TPS、延迟、瓶颈、水位。

延迟

latency / delay · *yán chí*

字　面　延，拖长；迟，晚到。日常汉语里多作动词：会议延迟半小时。

释　义　一次请求从发出到收到响应所经历的时间。工程上按分位数（P50、P99）而非平均值讨论，因为分布高度偏斜。

例　句　「跨地域一跳，延迟从 3ms 涨到 30ms，缓存不前置根本没法用。」

语言学注

语义窄化叠加词性漂移：通用语的「延迟」是动词性的「推迟发生」，术语义则名词化为一个可测量的连续量，于是「延迟是 30 毫秒」这种日常听来别扭的句子成了行话常态。英语区分 latency（固有耗时）与 delay（额外等待），中文一律折叠进「延迟」；同一概念电信业叫「时延」、定时任务叫「延时」，三个近形词按行业分片——语域即词形。初学者还要过一道「方向关」：延迟越低越好，「低延迟」是夸奖，与「高性能」的直觉方向相反。

关　联　尾延迟、长尾、抖动、吞吐。

长尾

`long tail` · *cháng wěi*

字　面　长长的尾巴。喻体是分布直方图的形状：头部高耸，右侧拖出一条又低又长的尾。

释　义　分布中远离主体、单个占比极小却总量可观的部分；性能语境特指最慢的那一小撮请求（长尾延迟）。

例　句　「均值挺漂亮，长尾吓死人——万分之一的请求要等三秒，投诉偏偏全来自这批人。」

语言学注

逐字仿译（long→长、tail→尾），且中文开发者先后从两条路遇到它：先是安德森《长尾理论》中译本炒热商业义，后来 tail latency 又把统计义带进性能圈——同一个仿译词、两次独立借入，两个义项在搜索引擎里至今互相干扰。它是典型的「图形寄生词」：词义寄生在一张分布曲线上，没见过直方图的人无从索解尾巴为什么代表慢请求；对老手则一词顶一张图，是极高效的沟通压缩。

关　联　尾延迟、毛刺、延迟。

热点／热键

`hotspot / hot key` · *rè diǎn ／ rè jiàn*

字　面　发热的点。物理意象：哪里被反复摩擦，哪里就发烫。

释　义　被访问得极不成比例地频繁的数据、代码路径或分区；「热键」特指缓存或分布式存储里被集中打爆的那一个 key。

例　句　「明星一官宣，Redis 里那个热键把整个分片打满，同分片的业务全跟着陪葬。」

语言学注

语义窄化的标本：通用语「热点」是舆论焦点，工程语义收窄为「访问频率异常集中的位置」。更值得注意的是背后整个概念隐喻的成建制移植——「访问频率是温度」：热数据／冷数据、预热、冷启动、降温，英语的 hot／cold 词族被整族仿译过来，学会一个就能推演一串。「热键」另藏同形陷阱：Windows 文档里 hotkey 译「热键」指快捷键，与缓存的「热键」同字同音不同义，新人跨子领域检索极易撞车。

关　联　冷启动、预热、长尾、水位。

冷启动

`cold start` · *lěng qǐ dòng*

字　面　机器冷着就点火。喻体来自内燃机：冬天冷车打火，费力、抖、动力差，得先热车。

释　义　系统在没有缓存、连接池、JIT 编译结果等「余温」的状态下从零启动，头几批请求明显偏慢；推荐系统里另指新用户、新物品无历史数据可用。

例　句　「函数计算是省钱，可冷启动一次一秒多，P99 全让它拖烂了。」

语言学注

仿译（cold→冷、start→启动），难得的是喻体跨语言毫发无损——热车经验中国司机同样熟悉，隐喻无需重新解释。真正的难点在「冷」的所指高度抽象：冷的不是温度，而是「缓存为空、连接未建、代码未编译」这一整包状态，一个形容词折叠了半页技术细节。同一词形还横跨三个子领域（Serverless 冷启动、App 冷启动、推荐冷启动），共享「无积累起步」的图式而细节迥异，初学者容易拿一个义项硬套另外两个。

关　联　预热、热点、延迟。

预热／热身／warm-up

`warm-up` · *yù rè ／ rè shēn*

字　面　预热，预先加热，本是锅炉、烤箱的工业用语；热身，运动前活动筋骨，来自体育。

释　义　在正式承接流量前主动加载缓存、建立连接、触发 JIT 编译，让系统进入「热」状态；也指对新扩容的节点缓慢放量。

例　句　「零点开抢前把热销 SKU 的缓存先预热一轮，别让第一波人全打穿到数据库。」

语言学注

同一个 warm-up，中文走了两条译路：书面文档用工业现成词「预热」（偏正式构词：预＋热），口语用体育词「热身」，还常拟人——「让服务先热热身」，把系统当运动员，语域亲昵。选哪个词能听出说话人的角色：写方案的人「预热」，蹲发布的人「热身」。理解难点在宾语省略：预热什么？缓存、JVM、连接池还是索引？动作的真正对象全藏在上下文里，新人听到「先预热再切流」往往不知道要热的是哪一层。

关　联　冷启动、热点、压测、扩容／缩容。

削峰填谷

`peak shaving / load shifting` · *xuē fēng tián gǔ*

字　面　削平山峰，填平山谷。喻体是土方与水利工程：把高处削下来的土，垫到低处去。

释　义　把流量高峰期承接不了的请求暂存起来（通常进消息队列），挪到低谷期慢慢处理，让负载曲线变平缓。

例　句　「秒杀那波流量别硬扛，前面加个消息队列削峰填谷，后端按自己的节奏慢慢消费。」

语言学注

本册四字并列格的极品：「削峰」「填谷」两个动宾结构工整相对——动词一削一填（一去一补），宾语一峰一谷（一凸一凹），完全套用「开源节流」「扬长避短」的成语铸词模板。文言骈俪的对仗在此不是修辞装饰，而是压缩算法：四个字装下识别峰谷、缓冲暂存、延迟消费、平滑负载一整套架构策略，展开够写一页设计文档。词源上它并非译词——先是电力行业的老术语（电网削峰填谷、峰谷电价），互联网整词借入，属跨行业迁移；英语反而没有对称的对应物，peak shaving 只削峰，填谷得另说 valley filling，合称 load shifting，对仗之美全无。初学者的障碍不在字面而在留白：词只描绘曲线愿景，不透露「怎么削」——排队、限流还是降级，全要另问。

关　联　限流、令牌桶、漏桶、水位。

压测／负载测试／压力测试

`load testing / stress testing` · *yā cè*

字　面　「压力测试」的紧缩。压，往上施加重量的物理隐喻。

释　义　用工具人为制造大流量打向系统，观察吞吐、延迟与错误率，摸清容量上限。严格说负载测试验证预期流量，压力测试打到崩溃为止，口语统称压测。

例　句　「上线前先压一轮，摸清单实例到底能扛多少，别到大促当天开盲盒。」

> **语言学注**
>
> 四音节术语缩合为双音节（压力测试→压测）是行话的标准工序（同「联调」「灰度」），更有趣的是下一步逆构：「压」从缩略词里重新析出为自由动词——「压一下」「再压十分钟」「压到两万 QPS」，能带宾语补语，行话里还派生出「摸高」（压到极限探顶）。围绕它的搭配整套武斗化：流量往上「打」，系统「扛」不「扛」得住，「顶」不住就「崩」——英语 load test 平铺直叙，中文语场却是武侠擂台，感情色彩浓烈。代价是精度：负载、压力、浸泡（soak）三种测试在口语里全被「压测」吞并，缩略词反噬了分类学。

关　联　基准测试、瓶颈、吞吐、削峰填谷。

基准测试／跑分／benchmark

`benchmark` · *jī zhǔn cè shì ／ pǎo fēn*

字　面　基准，比较用的标尺；英语 benchmark 原是测绘工匠凿在岩石上的水准点。跑分，跑出一个分数。

释　义　在受控、可复现的条件下测量性能，得到可横向比较的数字；跑分特指用现成套件刷分，带竞技色彩。

例　句　「口说无凭，跑个 benchmark——新旧版本各三轮，取中位数再下结论。」

> **语言学注**
>
> 一个概念三层语域三套词形，是意译、语码转换、俗译并存的活标本：文档写「基准测试」（意译，测绘词源在翻译中蒸发，只剩抽象的「基准」）；口语直接夹英文「跑个 benchmark」「bench 一下」；硬件与手机圈说「跑分」——动宾紧缩，自带安兔兔时代的刷榜竞技味。选词即身份：说「跑分」的像玩家，说「基准测试」的像写规范的人。初学者常把它与压测混为一谈，分界其实很硬：基准求可比（变量受控、结果可复现），压测求极限（越狠越好）。

关　联　压测、吞吐、QPS／TPS。

瓶颈

`bottleneck` · *píng jǐng*

字　面　瓶子的颈。整瓶水倒得再急，流速也被最细的那段管径卡死。

释　义　限制系统整体吞吐的那个最慢环节。定理内置：瓶颈不除，优化其余任何部分都不会更快。

例　句　「CPU 才三成、内存也闲着，吞吐就是上不去——瓶颈八成卡在数据库连接池。」

> **语言学注**
>
> 老资格仿译（bottle→瓶、neck→颈），进入汉语极早、泛化极深（人才瓶颈、发展瓶颈），以致工程语境是把一个已经磨平的日常词重新收紧、再术语化。这个隐喻的厉害在于自带推论：瓶子的物理结构决定流速只由最细处决定——「局部优化无用」这条反直觉的工程定律被免费打包在喻体里。汉语里它还与本土的「木桶短板」隐喻同场竞争，一个论流速、一个论容量，老手混用自如；新人则常犯「见慢就叫瓶颈」的错——按定义瓶颈同一时刻只有一个，消除一个才轮到下一个，这层串行约束就藏在瓶子的构造里。

关　联　吞吐、压测、热点、水位。

打点

`instrumentation / metrics reporting` · *dǎ diǎn*

字　面　打一个点。「打」是万能轻动词，「点」既是时间轴上的采样点，也是代码里的位置点。

释　义　在代码关键路径插入采集语句，向监控系统上报计数、耗时或事件；名词指所插的采集点本身。

例　句　「新接口记得打点，不然上线之后延迟涨没涨，心里完全没数。」

> **语言学注**
>
> 动宾紧缩的极简形态：轻动词「打」几乎不承担语义（同打车、打卡、打日志），全部信息压在一个「点」字上，两个字干完了英语 instrumentation 七个音节的活——好在英语新人同样看不懂 instrumentation，这对词算打了平手。中文特有的困扰是旧义干扰：「打点」在通用语里有打点行装、上下打点（疏通关系）等老义项，圈外人听到「上线前记得打点」难免误会要疏通什么关系。它与「埋点」的地盘至今没划清：大体打点偏后端指标、埋点偏前端行为，但互相越界成常态——一对近义术语的语义协商现在进行时。

关　联　埋点、水位、延迟、QPS／TPS。

埋点

`event tracking`（业内英文直译 `buried point`） · *mái diǎn*

字　面　把点埋进去。近乎埋设传感器：预先布下，静候有人踩中触发。

释　义　在产品界面或代码里预先植入行为采集逻辑，用户操作触发时向数据平台上报事件；增长与数据分析的地基。

例　句　「这个按钮的埋点漏报了三天，增长那边的转化漏斗直接没法看了。」

> **语言学注**
>
> 罕见的中文原生术语外销案例：国际通行说法是 event tracking，而中国公司英文文档里的 buried point 是从「埋点」逐字回译出去的中式英语——译借方向与本册几乎所有词条相反，术语发明权在中文互联网工业。与「打点」相比，「埋」多一层潜伏义：打是当下敲一记，埋是预先布设、延时触发；黑色幽默的用法里，漏埋错埋真的会被叫「埋了个雷」。词的派生力也证明它已扎根：无埋点、全埋点、可视化埋点——「埋点」本身成了能受否定和修饰的构词基座，这是外来概念完成本土化的形态学证据。

关　联　打点、水位。

水位／水位线／watermark

`watermark / high-low water mark` · *shuǐ wèi ／ shuǐ wèi xiàn*

字　面　水面高度。喻体来自水库大坝：标尺刻在坝体上，越过警戒线就要泄洪。

释　义　资源的当前占用程度（内存水位、集群水位、队列水位），常配高低水位阈值触发扩容或限流；流计算里的「水位线」另指事件时间推进到了哪里。

例　句　「集群水位到七成五了，要么赶紧扩容，要么大促当天就得限流保命。」

> **语言学注**
>
> 一个词形，三股来路：阈值义的高低水位线是借译——英语系统术语本就有 high-water mark（Kafka 的高水位、内核缓冲区水位）；「集群水位七成」这种占用率义却是中文顺着隐喻自己多长出来的一步，英语工程师更可能说 utilization——借来的隐喻在借入语里继续生长，是语言接触里很美的现象；Flink 的 watermark 译「水位线」是第三义，指事件时间戳，与资源占用毫无关系，新人在流计算文档里最容易在此翻车。再加一层感情色彩：对经历过汛期新闻叙事的中文使用者，「水位逼近警戒线」自带险情动员力，比中性的「利用率 85%」更能把人从工位上拽起来。

关　联　限流、削峰填谷、瓶颈、扩容／缩容。

令牌桶

`token bucket` · *lìng pái tǒng*

字　面　装令牌的桶。令牌是旧时官府军营的信物令箭：持牌者方可通行。

释　义　经典限流算法：系统以恒定速率向桶中投放令牌，请求须领到令牌才放行；桶有深度，攒下的令牌允许短时突发流量冲过去。

例　句　「网关按每秒一千往桶里放令牌，桶深给到两千，日常的小突发就让它冲。」

> **语言学注**
>
> 借译，但「token→令牌」是一次译名升格：英语 token 的本相是地铁代币、游戏筹码，市井而平凡；中文没译成「代币桶」，而是沿用令牌环网（Token Ring）时代定下的「令牌」——衙门信物的意象自带「凭证即权力」的威严，与「放行／拒绝」的语义严丝合缝，喻体反比原文传神。同一个 token 在中文里三世为人：网络与鉴权译「令牌」，编译原理译「记号／词元」，大模型时代干脆不译（「这段 prompt 多少 token」）——一词三译，恰是三个技术时代的地层剖面。「令牌桶」整体是纯虚构装置，现实中无人见过，初学者得先接受这个思想实验的设定，才谈得上理解算法。

关　联　漏桶、限流、削峰填谷。

漏桶

`leaky bucket` · *lòu tǒng*

字　面　底上带孔、匀速漏水的桶。来水再猛，出水永远是那一股细流。

释　义　与令牌桶齐名的限流整形算法：请求先入桶排队，以恒定速率流出处理，桶满则溢出丢弃。它彻底抹平突发，出口速率纹丝不动。

例　句　「对账接口对突发零容忍，套了个漏桶，出口速率死死钉在每秒两百。」

> **语言学注**
>
> 仿译（leaky→漏、bucket→桶），喻体家常到近乎寒酸，却是「隐喻即模型」的教科书：入水对到达流量、桶身对缓冲队列、漏孔对固定服务率、溢出对拒绝策略，物理部件与算法要素一一同构，记住这只桶等于记住整个算法。语言学上最妙的是褒贬倒置：「漏」在汉语里几乎从来是毛病（漏洞、泄漏、漏报），此处却是核心设计——新人第一反应「桶漏了不是 bug 吗」，老手知道漏得匀才是本事，一个字的感情色彩在语域切换中整个翻面。与令牌桶成镜像对子：一个管出水、一个管发牌，水利玩具模型的双璧。

关　联　令牌桶、限流、削峰填谷、背压。

QPS／TPS

`queries per second / transactions per second` · *字母词，无汉字读法，按英文字母名直读*

字　面　每秒查询数／每秒事务数。纯字母缩略，绕开汉字系统。

释　义　吞吐的两个标准计量词：QPS 数请求次数，TPS 数完整事务数。一次下单事务可能包含十几次查询，两者常被口语混用但量级迥异。

例　句　「日常八千 QPS、大促能冲五万，但真正的天花板是数据库那两千 TPS。」

> **语言学注**
>
> 字母词的典型标本：以拉丁字母形态直接嵌进汉语句法，前接数词（三万 QPS）、可作主宾语，却永远写不成汉字——「每秒查询数」是释义，不是可说的词。这暴露了汉语构词的一个结构洞：缺少英语首字母缩略那样高产的缩合机制，凡计量、指标类概念最易整形借入。社会语言学上它是行业切口式的身份暗号：「你们系统多大 QPS」既是技术提问也是黑话握手，答得出数量级才算圈内人。新人的门槛因此是双重的：先解开缩写，还要建立「多少算大」的行情感——八千 QPS 是大是小，取决于读写比与业务形态，数字素养藏在词后面。

关　联　吞吐、压测、尾延迟、打点。

尾延迟／P99／P999

`tail latency / 99th & 99.9th percentile` · *wěi yán chí*

字　面　尾部的延迟。P99 即第 99 百分位：把所有请求按快慢排序，第 99% 位置那一个的耗时。

释　义　延迟分布最慢一端的水平：P99 意味着 99% 的请求比它快；P999 再苛刻十倍。SLO 通常钉在尾延迟而非均值上，因为用户记住的是最慢的那次。

例　句　「别拿均值糊弄人，SLO 写的是 P99 小于 100 毫秒，超一点都算违约。」

> **语言学注**
>
> 混合构词的展台：「尾延迟」是 tail latency 的仿译，「尾」直接续接「长尾」的曲线隐喻；P99 则是字母加数字的记号词，连读法都还没在中文里定型——「P 九九」「P 九十九」「三个九」并存，一个词的语音形式仍在野蛮生长，是观察术语标准化过程的活切片。它最深的门槛不在语言而在统计：新人常把「P99 等于 100 毫秒」误读成「99% 的时间延迟是 100 毫秒」，分位数概念没装进脑子，词形再透明也没用——术语难懂的第三种原因，构词学到此让位给数学。

关　联　长尾、延迟、毛刺、QPS／TPS。

毛刺

`spike / glitch` · *máo cì*

字　面　金属或木料切削后残留在棱边的细小尖刺。机加工车间用语：工件出活前要去毛刺、打磨。

释　义　监控曲线上突然蹿起又迅速回落的孤立尖峰，多指延迟或错误率的瞬时异常；幅度不大、来去无踪的那种最磨人。

例　句　「每天凌晨三点延迟曲线准时冒一根毛刺，查了半个月，是日志轮转干的。」

> **语言学注**
>
> 本土隐喻的自发生成，不是译词：英语说 spike（尖桩）或 blip，中文没有仿译成「尖钉」，而是从制造业车间调来了「毛刺」。「毛」言其细碎不规则，「刺」言其扎手，两个语素合力传递「小而恼人」的质感，还比 spike 多一层价值判断——毛刺是工艺不完美的残留物，理应打磨掉，这暗合监控文化的审美：曲线要「干净」「平滑」，有毛刺是丢人的。与「长尾」同属图形寄生词，词义长在 Grafana 的曲线上，没盯过大盘的人不知道刺从何来；天天盯盘的人则一个字就能指认现场。

关　联　抖动、长尾、尾延迟。

抖动

`jitter` · *dǒu dòng*

字　面　发抖、颤动，身体或机械的高频小幅振颤。

释　义　指标围绕基线的高频小幅波动。网络工程里 jitter 有严格定义（包间延迟之差），互联网口语里泛化为一切不稳——性能抖动、服务抖动、网络抖一下。

例　句　「专线均值延迟不高，就是抖动大，视频会议隔几秒糊一下。」

> **语言学注**
>
> 意译顺滑（jitter 与「抖」都从神经质的颤动引申），值得看的是它与「毛刺」形成的民间形态学：毛刺是孤立的一根刺，抖动是连续的颤——工程师用两个隐喻词给曲线形态做了分类，这种精确分工日常汉语里并不存在，属于借入后行话内部的语义再精密化。另有拟人用法「服务在抖」「网络一抖，重试风暴就来了」，把系统当成受惊的活物，语域亲昵而带宿命感。新人的坑在精度落差：网工的 jitter 有公式可算，黑话的「抖动」只是模糊体感，同一词跨语域精度衰减一个量级。

关　联　毛刺、延迟、尾延迟、卡顿。

卡顿

lag / jank / stutter · *kǎ dùn*

字　面　卡，卡住、卡壳；顿，停顿。两个「停滞」义语素近义并列。

释　义　画面或交互间歇性停滞的用户侧体感——帧率不稳、主线程阻塞、滑动掉帧；移动端与游戏语境的核心性能词，并已量化为「卡顿率」等正式指标。

例　句　「新版列表页滑动有点卡顿，一查，有人在主线程里解析大 JSON。」

> **语言学注**
>
> 本册罕见的「体感优先」词：吞吐、P99 都站在机器一侧测量，「卡顿」站在人这一侧描述知觉，而且路径自下而上——先在玩家口语里生长（好卡、卡成 PPT），再被工程界收编成大盘指标名，是俗词术语化，与多数词条的术语俗化方向相反。构词是近义联合（卡＋顿），单字「卡」的派生活力极强：能重叠（一卡一卡的）、能作补语（卡死、卡爆）、能程度化（有点卡、卡麻了）。英语至今没有统一对应词，lag、jank、stutter、freeze 各据一方，中文反而先有了稳定统称——一个译不出去的本土好词。

关　联　延迟、抖动、毛刺、冷启动。

扩容／缩容

scale out / scale in（弹性伸缩 **auto-scaling**） · *kuò róng / suō róng*

字　面　扩大／收缩容量。容，容积、容纳，上承「容量规划」「容灾」的容字族。

释　义　增减系统资源规模：横向加减机器、纵向升降配置；交给策略自动执行就是弹性伸缩。

例　句　「大促前手动扩到三倍，活动一停连夜缩容，不然月底账单没法看。」

> **语言学注**
>
> 一对靠替换单个反义语素铸成的对称双子词，汉语单字反义系统的构词经济性尽显：扩／缩一换，方向反转，其余全部复用。英语这边却要 scale out、scale in、scale up、scale down 四个短语，还常年争论 out 与 up 之别；中文两个双音词全收，代价是横向纵向之分被抹掉，要说清得补「横向扩容」「垂直扩容」。语域上另有一套内部白话：「加机器」「砍机器」，直白得不像术语——对客户讲扩容，对同事讲加机器，一个概念两副面孔，对外术语与对内白话的双层词汇是行话社会学的常态。

关　联　水位、削峰填谷、压测、冷启动。

第六辑 · 数据与并发

多份数据、多条线程如何各自为政又勉强一致——从「落盘」到「脑裂」。

分片

sharding / shard · *fēn piàn*

字　面　分＝切分，片＝薄片、碎块。英文 shard 本义是打碎的陶片，喻体来自「完整器物碎成小块」的场景。

释　义　把一个大数据集按分片键水平切成多份，分散到多台机器存储与查询，突破单机容量与吞吐上限；切出来的每一份也叫「一个分片」。

例　句　「用户表按 user_id 哈希分片，一共 64 个分片，扩容还得重新搬数据。」

> **语言学注**
>
> 意译兼弱化的仿译：shard 的「碎陶片」意象在中文里被磨平，「片」只剩「切出的一小块」这一近乎透明的量词式语义，破碎感消失、秩序感增强——中译悄悄改写了隐喻的情绪。构词上「分片」动名兼类：「这张表要分片」（动词）、「第 3 个分片挂了」（名词），是双音节技术词的典型行为。newcomer 的难点不在字面而在近义词丛：分片、分区（partition）、分库分表所指部分重叠，英文社区同样混用，翻译如实继承了这笔糊涂账。

关　联　分库分表、读写分离、主从／主备

分库分表

database & table sharding（中文自造总称） · *fēn kù fēn biǎo*

字　面　把数据库分开、把数据表分开，两个同构动宾短语的叠加。

释　义　中文圈对关系库水平拆分的习用总称：按业务或分片键把一个库拆成多库（分库）、一张表拆成多表（分表），常配中间件做路由与聚合。

例　句　「订单表过亿了，先分表看看，扛不住再分库，中间件就用 ShardingSphere。」

语言学注

四字并列格：「分库」「分表」2+2 音步对举，念起来像成语，庄重且顺口。更值得记录的是它基本是中文社区自造的本土术语——英文只说 sharding／partitioning，并无逐字对应词——是汉语构词力反哺技术话语的少数样本。四字格的紧凑也带来误导：字面像一套连招，实际「只分表不分库」「只分库不分表」都很常见，组合自由度被并列格式掩盖；面试语境里它还常与「读写分离」连用成八字套餐，仪式感大于信息量。

关　联　分片、读写分离、主从／主备

读写分离

`read/write splitting` · *dú xiě fēn lí*

字　面　把「读」与「写」分开处理。

释　义　写请求只走主库，读请求走从库或只读副本，用复制换读吞吐；代价是主从延迟——刚写进去的数据可能一时读不到。

例　句　「上了读写分离之后，下完单立刻查订单可能查不到，前端记得做个兜底重试。」

语言学注

四字并列格，且语体向政策文言靠拢——与「政企分开」「政教分离」同构，自带制度设计的庄重感，与其「架构层约定」的实质相称。整体高度名词化：口语说「上读写分离」「做读写分离」，四字格充当宾语，动词另配。newcomer 的两个坑都在字面之外：其一，它不是数据库自带开关而是路由约定；其二，「分离」暗示切得干净，掩盖了「读己之写」等一致性毛边——术语的整齐反而遮蔽了系统的不整齐。

关　联　主从／主备、分库分表、最终一致性、写扩散／读扩散

主从／主备（主副本-从副本）

`master/slave, primary/replica, active/standby` · *zhǔ cóng ／ zhǔ bèi*

字　面　主＝主人、为主者；从＝仆从、跟随者；备＝后备、备用。喻体是传统的主仆尊卑与正备编制。

释　义　一种复制拓扑：主节点接受写入，从节点持续复制主节点的变更，用于分担读或待命接管。「主从」偏重从库参与服务，「主备」偏重备节点平时不接流量、只等接管。

例　句　「主从延迟都 3 秒了，报表那些重查询先切到从库，别再打主库。」

语言学注

仿译 master/slave，但两种语言背负的历史完全不同：英文因奴隶制联想在 2010 年代后系统性改用 primary/replica、leader/follower；中文「主从」的喻体是本土主仆关系，刺激性弱得多，几乎没有更名压力——同一对概念在两个语言社区的感情色彩负担截然不同，是术语「政治语义学」的绝佳标本（中文文档改用「主副本／从副本」多是翻译同步，并非本土诉求）。「主备」与「主从」一字之差编码了「备节点是否对外服务」这一架构事实，这种最小对立正是行话信息密度之所在。

关　联　读写分离、心跳、脑裂、强一致性

回源

`back-to-origin fetch` · *huí yuán*

字　面　回到源头去。源＝源站，即被 CDN／缓存层挡在身后的原始服务器。

释　义　缓存未命中或已过期时，缓存节点向源站请求数据的动作；「回源率」是衡量缓存命中效果的核心反向指标。

例　句　「昨晚边缘节点批量重启，回源率一下飙到 40%，源站差点被打挂。」

语言学注

动宾紧缩的标本级案例：「（缓存未命中，节点）回（到）源（站取数据）」——施事、条件、目的全部省略，只留一个运动方向，两个字扛起一整个事件脚本。对圈内人一字千金，对圈外人零线索：「源」指什么（水源？电源？源码？）字面毫无提示，必须先共享「CDN—源站」这幅世界图景才能解码——紧缩词省掉的从来不是信息，而是「默认你已知道」的共识。站稳之后它还获得类词根的能产性：回源率、回源超时、防穿透回源，一词生一族。

关　联　缓存穿透／缓存击穿／缓存雪崩、惊群、落盘

落盘

`persist to disk` · *luò pán*

字　面　落到磁盘上。落＝下落、着地；盘＝磁盘。

释　义　数据从易失的内存真正写入持久化介质、掉电不丢的那个状态跃迁；强调「已持久化」的结果，而非「执行了写操作」的过程。

例　句　「broker 返回 ack 之前消息必须落盘，不然一重启这批数据就没了。」

> **语言学注**
>
> 动宾（趋近动补）紧缩兼隐喻引申：「落」自带重力意象——数据在内存里是悬着的、飘着的，落到盘上才踏实，一个字同时编码方向、终态与安全感，这层意象英文 persist／flush 里没有。与「写盘」的差别是行话的精微处：「写」只陈述动作（可能还停在 OS 页缓存里），「落」断言结果，所以老手说「写了不等于落了」。newcomer 的困惑在于它听着像单一物理动作，实际是穿过应用缓冲、page cache、磁盘缓存多层「谎言」之后才成立的抽象承诺——一个字的笃定，背后是一列 `fsync` 的辛酸。

关　联　刷盘、脏页、双写

刷盘

`flush (to disk)` · *shuā pán*

字　面　把缓冲区「刷」到磁盘上。刷＝用刷子清扫、快速掠过。

释　义　主动把内存中攒下的脏数据批量写入磁盘的动作，对应 flush／`fsync`；常见搭配「同步刷盘／异步刷盘」「刷盘策略」「刷盘时机」。

例　句　「生产环境别开同步刷盘，延迟顶不住；异步刷盘加上主从复制，可靠性够用了。」

> **语言学注**
>
> 动宾紧缩＋喻体置换的仿译：flush 本义是「冲水」（马桶、管道），中文没有沿用水流意象，改选「刷」——从「冲走」变成「刷过」，动作更主动、更有清扫感，还与「刷墙」「刷卡」「刷屏」共享「快速批量掠过」的现代义项。它与「落盘」构成动作／结果的最小对立：刷盘是我发起的行为，落盘是数据到达的状态；「刷了盘未必落了盘」（盘上还有硬件缓存）这句绕口令式辨析，恰好暴露两个紧缩词各自锚定的语义焦点——外人听来是同义反复，内行听来是两层保证。

关　联　落盘、脏页、优雅关闭／优雅退出

脏页

`dirty page` · *zāng yè*

字　面　脏的内存页。页＝操作系统分页；脏＝被弄脏、待清洗。

释　义　内存中已被修改、尚未写回磁盘的页，与磁盘副本不一致，须由回写机制择机刷盘；积压过多会触发集中回写、拖垮 IO。

例　句　「IO 突然打满，一查是脏页水位过线，内核集中回写把带宽吃光了。」

> **语言学注**
>
> 仿译 dirty page，是「贬义词术语化去贬义」的教科书案例：「脏」从道德／卫生域被征用为纯技术状态词——脏页不是坏页，是系统正常工作的痕迹，感情色彩被彻底漂白。newcomer 恰恰栽在这里：听到「脏」以为出了故障，老手只把它读作「未落盘」三个字。「脏」由此成为能产语素，繁殖出脏读、脏写、脏数据、脏检查（dirty check）一整族——一次成功的隐喻移植，回报是一个构词位。与「干净页」（clean page）的显式反义配对，则把「洗净＝同步到盘」的隐喻闭环补全了。

关　联　刷盘、落盘、脏读

快照隔离

`snapshot isolation` · *kuài zhào gé lí*

字　面　快照＝快速拍下的照片；隔离＝彼此隔开。每个事务对着自己那张「照片」干活。

释　义　一种事务隔离实现：事务开始时取得数据库的一致性快照，全程读这份快照、与并发写互不阻塞，提交时才检测写冲突；MVCC 是其典型底座。

例　句　「PostgreSQL 的 Repeatable Read 实际就是快照隔离，写偏斜它防不住，得上 Serializable。」

> **语言学注**
>
> 两个仿译的叠加：snapshot→快照（摄影喻），isolation→隔离（检疫喻）。摄影喻的成功在于精准传达「时点冻结，之后世界的变化与我无关」；但隐喻总有盲区——照片本该只读，快照隔离却允许你「往照片上写字」再合并回去，写偏斜（write skew）异常恰恰藏在喻体覆盖不到的褶皱里。另一笔翻译遗产是「隔离级别」的「级别」：它暗示一条线性阶梯，而真实的隔离模型是偏序关系，快照隔离与可重复读谁高谁低说不清——被「级别」二字喂大的直觉，日后要用事故偿还。

关　联　脏读、不可重复读、幻读、写时复制

乐观锁

`optimistic lock(ing)` · *lè guān suǒ*

字　面　乐观的锁——把一种人生态度安到并发控制上。

释　义　并不真正加锁的冲突控制：假设冲突罕见，先直接干活，提交时用版本号／CAS 校验数据没被人动过，动过就重试或报错。读多写少场景的首选。

例　句　「库存扣减用乐观锁就行，update 带上 version，影响行数是 0 就重试两次。」

> **语言学注**
>
> 仿译 optimistic locking 的拟人隐喻：把「对冲突概率的假设」写成「性格」——乐观者先做后查，悲观者先锁后做，一个形容词连选型条件（冲突多不多）都教给你了，是隐喻承担教学功能的范例。但它同时名不副实：乐观锁根本没有锁，是「无锁＋提交校验」，「锁」在此已语义漂白为「并发控制手段」的类后缀——newcomer 追问「乐观锁到底锁住了什么」，就是被词形骗进了这个词汇化陷阱；老手的标准玩笑「它锁了个寂寞」，又把陷阱变成了接头暗号。

关　联　悲观锁、自旋锁、幂等键

悲观锁

`pessimistic lock(ing)` · *bēi guān suǒ*

字　面　悲观的锁。

释　义　假设冲突随时会发生，动手前先真正加锁（如 `SELECT ... FOR UPDATE`），把并发串行化；换来确定性，付出吞吐和死锁风险。

例　句　「对账这段绝对不能并发，直接 for update 上悲观锁，慢点就慢点，图个安稳。」

> **语言学注**
>
> 与「乐观锁」构成拟人隐喻的对偶：汉语借单音节反义语素（乐／悲）造出只差一字的最小对立对，比英文 optimistic／pessimistic 的多音节对峙轻快得多——工程术语爱成对（强／弱、读／写、主／从），而汉语的单字反义库让配对格外整齐，这是汉语在术语系统性上的结构红利。感情色彩上「悲观」在此无贬义，语用中反而常带「稳妥、保命」的褒义（「悲观锁虽慢，晚上睡得着」）——形容词的词典义与行话语用义在这里分了家。

关　联　乐观锁、读写锁、死锁／活锁

自旋锁

`spinlock` · *zì xuán suǒ*

字　面　自己原地旋转的锁。自旋＝原地打转。

释　义　拿不到锁时不睡眠、不让出 CPU，在循环里反复尝试（忙等）；临界区极短时比挂起唤醒便宜，转久了就是纯烧 CPU。

例　句　「临界区就改两个指针，用自旋锁划算，犯不着让线程陷到内核里睡一觉。」

> **语言学注**
>
> 仿译 spinlock。「自旋」撞上了物理学既有译名（电子自旋 spin），属跨学科词形复用——懂点物理的初学者反而容易被量子意象带偏，以为多高深，其实就是「原地转圈等」。「自」字是这个译名的点睛处：它标记忙等是线程自己主动的选择，而非被调度器挂起，一字之差分开了两种等待哲学（spin vs block）。口语中「在那儿自旋」「空转」已回收为普通动词——双音节译词扎根之后的动词化，是行话融入日常语法的常规路径。

关　联　悲观锁、读写锁、惊群

读写锁

`read-write lock` · *dú xiě suǒ*

字　面　区分「读」与「写」的锁。

释　义　一把锁的两种获取模式：读锁共享、写锁排他——读读并行、读写互斥、写写互斥；适合读远多于写的数据结构。

例　句　「配置热更新那块换读写锁吧，读路径全在 mutex 上排队，白白浪费。」

> **语言学注**
>
> 「读写」并列语素＋中心语「锁」的偏正结构，与「读写分离」共享前件——同一语素组在锁粒度与架构拓扑两个层级复用，听者须靠中心语切换语义框架，这是行话省力原则的直接体现。规则矩阵则被压进「读读并行、读写互斥、写写互斥」的四字口诀链，骈俪节奏直接当记忆编码用。newcomer 常按字面把它当成「一把管读的锁＋一把管写的锁」，实际是同一把锁的模式之分；「写锁饥饿」等衍生词还要求理解两种模式的优先级博弈——字面给的越少，上下文欠的越多。

关　联　读写分离、悲观锁、自旋锁

心跳

`heartbeat` · *xīn tiào*

字　面　心脏的跳动，生命体征的代名词。

释　义　节点间周期性发送的「我还活着」信号；连续多个周期收不到，即判定对端故障，触发摘流量、重选主等动作。

例　句　「session 超时被 ZK 摘了，八成是 GC 停顿太长，心跳没发出去。」

语言学注

教科书级仿译（calque）：heartbeat→心跳，喻体是「以心跳判生死」的监护经验，人类共通，跨语言零损耗——所以它是分布式词汇里几乎唯一圈外人也能秒懂的词。行话深度全在派生搭配：丢心跳、心跳风暴、心跳超时；以及隐喻掩盖的认识论陷阱——收不到心跳只证明「联系不上」，不证明「死了」，把「无心跳」直接当「死亡」正是脑裂事故的第一块多米诺。一个完美的隐喻，恰恰因为太好懂，让人忘了它只是隐喻。

关　联　探活、保活、脑裂、主从／主备

探活（存活探针／就绪探针）

`liveness probe / readiness probe` · *tàn huó*

字　面　探一探还活着没有。探＝主动查看、试探；活＝存活。

释　义　由外部主体（负载均衡、K8s 的 kubelet）周期性主动检查服务健康：存活探针不过关就重启容器，就绪探针不过关就摘掉流量。口语统称「探活」。

例　句　「先把探活接口写了再上线，不然 readiness 一直不过，流量根本进不来 pod。」

语言学注

动宾紧缩：「探」＋「活」两字压缩了「主动发起探测以确认对方存活」的完整施事结构。它与「心跳」构成信息流向的最小对立系统：心跳是被监控者主动上报（push），探活是监控者上门查问（pull）——各锚定一个方向，老手听词即知架构长相，newcomer 却常把两者混作一团。正式语体用仿译「探针」（probe，医疗器械喻体），口语几乎只说紧缩式「探活」——同一概念在文档语与工程口语中分层共存；「活」还串起探活、保活一族，构成生命隐喻的小词场。

关　联　心跳、保活、优雅关闭／优雅退出

保活

`keep-alive` · *bǎo huó*

字　面　保住性命、维持存活。

释　义　靠周期性空报文或协议机制让一条连接（TCP／HTTP keep-alive）不被对端或中间设备超时回收；移动端语境又指让后台进程不被系统杀掉。

例　句　「网关到后端连接池开着 keep-alive 呢，别每个请求都重新握手，白多一个 RTT。」

语言学注

动宾紧缩＋逐字仿译（keep→保，alive→活）。「保」在汉语里自带「保温、保鲜、保胎」的维持类构词框架，母语者几乎能自解释；与「探活」共享语素「活」却施事相反——保活保的是自己（这条连接／这个进程）别被判死，探活探的是别人死没死，一守一攻，同一个「活」字两副面孔。真正的难点是所指多义：TCP 保活、HTTP 连接复用、安卓进程保活是三件颇为不同的事，共用一个词，跨子领域交流时常常各说各的「保活」——紧缩词的省力，以歧义为利息。

关　联　探活、心跳、优雅关闭／优雅退出

优雅关闭／优雅退出

`graceful shutdown / graceful exit` · *yōu yǎ guān bì / yōu yǎ tuì chū*

字　面　优雅地关门退场。优雅＝仪态从容、不失体面。

释　义　进程收到终止信号后不立即退出：先摘流量、拒新请求、把在途请求处理完、落盘该落的数据、释放资源，然后才退出，保证不丢数据不断连接。

例　句　「发布一抖一片 502，就是没做优雅关闭，pod 说杀就杀，在途请求全被掐了。」

语言学注

仿译 graceful shutdown 的语域反差样本：graceful 在英文工程语境早已漂白成「有序、不粗暴」，中文却选了美学浓度极高的「优雅」，把描写仪态的词安在进程之死上，初听近乎滑稽——进程还讲优雅？竞争译名「平滑关闭」「有序退出」更准确却都没赢，说明术语传播偏爱有画面、带一点幽默的译法。它的反义概念在口语里叫「暴力杀」「`kill -9` 干掉」——「优雅／暴力」这对日常形容词，就此把 SIGTERM 与 SIGKILL 的信号语义整个装了进去，感情色彩替技术细节站岗。

关　联　探活、保活、落盘

脏读

`dirty read` · *zāng dú*

字　面　读到了「脏」东西的一次读取。

释　义　事务 A 读到事务 B 已修改但尚未提交的数据；若 B 随后回滚，A 读到的就是一条从未真正存在过的数据。只有最低隔离级别（读未提交）才放任它。

例　句　「余额一刷一个数，查半天原来隔离级别被设成读未提交了，妥妥的脏读。」

语言学注

仿译 dirty read。「脏」与「脏页」同源却已语义分岔：脏页之脏＝改了没刷盘（相对磁盘不一致），脏读之脏＝没提交（相对事务语义不作数）——同一隐喻语素在 OS 层与数据库层各自窄化，newcomer 极易串台，这是隐喻能产性的代价。构词上是「形＋动」转指名词的紧缩：读的并不是「脏」这个东西，而是「这次读被污染了」，论元关系被两字格压得全无踪影。三大读异常里它最好学，因为「脏」自带「来路不干净」的道德直觉——隐喻替定义打了前站。

关　联　幻读、不可重复读、脏页

幻读

`phantom read` · *huàn dú*

字　面　见了幻影的一次读取。幻＝虚幻、幻觉。

释　义　同一事务内用相同条件先后两次范围查询，第二次凭空多出几行——别的事务插入并提交的「幻影行」。与不可重复读的分界：幻读是行的集合变了，不可重复读是同一行的值变了。

例　句　「先 count 再逐条处理，数对不上，中间有人插了新订单——典型幻读，上间隙锁或者干脆串行化。」

语言学注

仿译 phantom read，但喻体动了手术：phantom 在西方语境是幽灵、显形的亡魂，中文弃「鬼」取「幻」，从灵异域挪到感知域（幻觉、幻影），既去了迷信味，也更贴合「两次快照对不上」的主观体验——是喻体本土化改造的漂亮案例。风险在于隐喻方向容易记反：不少新人以为幻读是「读到了不存在的数据」（那更像脏读），实际是「新出现的行打破了你以为封闭的世界」——隐喻给了记忆钩子，却不给语义边界，三兄弟必须对照着学，单独背哪个都会跑偏。

关　联　脏读、不可重复读、快照隔离

不可重复读

`non-repeatable read` · *bù kě chóng fù dú*

字　面　无法重复的读取——同一事务里同一条查询，两次结果不一样。

释　义　事务 A 两次读同一行，中间事务 B 修改（或删除）该行并提交，A 前后读到的值不一致。盯的是既有行的值变化；行的增减归幻读管。

例　句　「Repeatable Read 顾名思义保证的就是可重复读，MySQL 在这一级还顺手挡掉了大半幻读。」

语言学注

三大读异常里唯一没有隐喻的成员：脏读、幻读各凭一个形容词隐喻一字定调，它却是「不可＋重复＋读」的否定式描写性定名——五个音节，拗口而精确。这种隐喻词与描写词的混编，恰好摆出术语翻译的两难：隐喻好记但边界含糊，描写精确但冗长无画面。经验上初学者最记不住的恰恰是它——记忆负担与意象缺失直接相关，可当词汇教学的现成实验组；老手则靠它的否定形反推正名（Repeatable Read），词形长反而成了锚点。

关　联　脏读、幻读、快照隔离

生产者-消费者

`producer-consumer` · *shēng chǎn zhě – xiāo fèi zhě*

字　面　生产货物的人与消费货物的人，喻体来自经济学的市场角色对。

释　义　经典并发／消息模式：一方产出数据投入缓冲队列，另一方取出处理，两端速率解耦；也是 Kafka 等消息系统的 API 词汇（producer／consumer）。

例　句　「消费者组又扩了两台机器，消费速度还是追不上生产，积压 200 万条了。」

语言学注

借译加语义窄化的双层样本：producer／consumer 先在英语内部从经济学借入 CS，中文再仿译进来。关键窄化发生在「消费」——日常义是花钱购物，队列语境里窄化为「取走并处理一条消息」，货币语义彻底蒸发。于是「重复消费」在商场是拉动内需、在消息系统是资损事故，「消费不动了」是机器慢而不是没钱——圈外人对这些搭配全无解码入口。文言后缀「-者」在口语中常被替换成拓扑方位词「生产端／消费端」，拟人角色悄悄退回机器身份，是行话去修辞化的常见轨迹。

关　联　扇入／扇出、背压、幂等键

幂等键

`idempotency key` · *mì děng jiàn*

字　面　幂等＝幂运算结果相等（数学定译，f(f(x))=f(x)）；键＝唯一标识。保证幂等性的那把「钥匙」。

释　义　客户端为一次业务操作生成的唯一标识；服务端据此去重，同一个键无论重放多少次，效果都等于执行一次。支付、下单等重试敏感接口的标配。

例　句　「超时是会自动重试的，扣款接口必须带幂等键，不然用户被扣两次钱，你就上热搜了。」

语言学注

罕见的「数学定译整包平移」：idempotent 源自拉丁语 idem（相同）＋potens（能、幂），中文数学界早有「幂等」定译，工程界直接继承，于是一个抽象代数词出现在支付对账群聊里，语域跳跃拉满。对初学者是三重门槛：「幂」字本身不透明、数学定义抽象、工程义还要再引申为「重试安全」；老手却离不开它——「这接口幂等吗」五个字顶一页容错需求文档，是行话压缩率的天花板之一。「键」则是 key 的成熟仿译，与主键、外键同族，落地毫无阻力。

关　联　双写、最终一致性、生产者-消费者

最终一致性

`eventual consistency` · *zuì zhōng yī zhì xìng*

字　面　最终（总会）一致。眼下允许不一致，假以时日必然收敛。

释　义　分布式副本间允许暂时读到旧值；停止写入后，经有限时间所有副本收敛到同一状态。可用性优先阵营的标准一致性承诺。

例　句　「点赞数各端对不齐很正常，这条链路本来就是最终一致性，别当资损问题提工单。」

语言学注

仿译 eventual consistency，难点全在 eventual：英文是「早晚会发生，但不承诺何时」的弱担保，中文「最终」却常被读出「终局必达」的强语气，译名比原文听起来更像承诺——于是社区流传自嘲式补丁「最终一致性，就是现在不一致」，一句玩笑替译名找补回丢失的语气。更深的坑是「一致性」的同形冲突：ACID 的 C、CAP 的 C、缓存一致性的 consistency 所指各不相同，全被同一个中文词统收——newcomer 大量「懂了又没懂」的挫败，病根在词典而不在教材。

关　联　强一致性、主从／主备、双写

强一致性

`strong consistency` · *qiáng yī zhì xìng*

字　面　强的一致性——保证更硬、约束更严。

释　义　任何读都能看到最近一次成功写入（严格形式即线性一致性）；副本分歧对外不可见，代价是延迟升高与分区时的可用性受损。

例　句　「库存链路要强一致性，读缓存都不行，要么直连主库要么上 Raft。」

语言学注

「强／弱」是学术翻译里最省事也最磨损的形容词对（强类型、强引用、强一致）：此处的「强」不是力气大，而是数学「strong／weak condition」的序关系标记——约束更严、能推出的保证更多——这层用法对没受过形式化训练的人是隐形门槛。与「最终一致性」的并置还有修辞不对称：一个以强度命名，一个以时间命名，本不在同一维度，口语却把它们当两极开关用——真实的一致性模型是一整张谱系（线性、顺序、因果、会话……），二元话语抹平了谱系，也预定了架构评审里的各说各话。

关　联　最终一致性、主从／主备、脑裂

写扩散／读扩散

`fan-out on write / fan-out on read` · *xiě kuò sàn / dú kuò sàn*

字　面　写入时向外扩散／读取时向外扩散。扩散＝像气味、涟漪般向四周散开。

释　义　Feed 流一对多分发的两种策略：写扩散在发布时把内容推进每个粉丝的收件箱（写重读轻），读扩散在拉取时实时聚合所有关注对象的发件箱（写轻读重）；大 V 场景常两者混合。

例　句　「千万粉的大 V 不能走写扩散，一条动态要写一千万个收件箱，给他切读扩散加缓存吧。」

语言学注

对 fan-out on write／read 的意译再造：舍弃机械感的「扇」，改用物理化学的「扩散」，多了一层弥漫开来、难以收拾的预警意味——「写扩散」三个字自带成本失控的气息。构词是状中紧缩：「写时扩散」省掉「时」，把条件从句压进双字格，「on write」的条件语义从字面彻底消失——不懂行的人无法从「写扩散」还原出「在写入那一刻」，再次印证紧缩词省掉的都是圈内共识。作为微博式 Feed 流工程孕育的词，它已成为中文面试黑话的核心考点，术语流通度与本土业务史深度绑定。

关　联　扇入／扇出、双写、生产者-消费者

双写

`dual write / double write` · *shuāng xiě*

字　面　写两份、往两处写。

释　义　同一份数据同步写往两个存储（数据库＋缓存、旧库＋新库）。数据迁移与冗余的常用过渡手段，也是一致性事故高发词：两次写之间没有事务，必然存在不一致窗口。

例　句　「先双写三周，对账脚本跑到零差异，再把读流量灰度切到新库，最后停旧库。」

语言学注

数词＋动词的极限紧缩（同族有双活、双发、三副本）：宾语、目的地、时序全部省略，字面信息量几乎为零，圈内却能自动展开成一整套迁移剧本——双写、对账、灰度切读、下线旧库，四幕连台。它是本册「一字千金而外人不明所以」的极致标本：两个字的解码需要一整段职业经历垫底。感情色彩上，方案评审里的「又要双写？」几乎总带警惕的贬义——老手都被双写不一致坑过，词的情感包浆来自集体事故记忆，这种由创伤积累的语用色彩，词典永远来不及收录。

关　联　写扩散／读扩散、幂等键、最终一致性

扇入／扇出

fan-in / fan-out · *shàn rù / shàn chū*

字　面　像折扇一样收拢进来／张开出去。喻体是折扇的几何形状：一个轴心连着一圈扇骨。

释　义　一个节点汇聚多个上游叫扇入，一个节点分发给多个下游叫扇出；源自数字电路（逻辑门的输入／输出端数），今泛用于调用拓扑、消息分发与并发编程。「扇出比」衡量一次请求触发多少次下游调用。

例　句　「入口一个请求扇出二十多个 RPC，尾延迟根本压不住，先把扇出砍一半。」

语言学注

仿译 fan-in／fan-out，且是工程内部的二次借用（数字电路→分布式）。「扇」的几何直观极好：从轴到边的辐射形状精确对应一对多拓扑。但它藏着一个汉语特有的解码事故：「扇」多音——名词 shàn（扇子）与动词 shān（扇风、扇耳光）——被读成动词时方向感当场错乱（是我扇别人还是别人扇我？），名词喻体被误读为动作，是单字隐喻独有的坑。它与本土再造的「写扩散／读扩散」并存竞争：前者通用而中性，后者场景化且自带成本语感——一对概念两套词，是外来仿译与本土意译分工共存的活标本。

关　联　写扩散／读扩散、生产者-消费者、背压

脑裂

split-brain · *nǎo liè*

字　面　大脑裂成两半。喻体来自神经医学：切断胼胝体后左右脑各自为政的「裂脑」病症。

释　义　网络分区把集群断成两半，双方都联系不上对方、都以为自己该当家，各自选主继续写入——出现两个「主」，数据从此分叉。多数派仲裁（quorum）就是为它准备的疫苗。

例　句　「两机房之间光纤一断，两边各选了个主，脑裂了，恢复之后对账对了一整周。」

> **语言学注**
>
> 仿译 split-brain，本册最惊悚的隐喻：把集群拟作一具有机体，裂开的不是网线而是「脑」——喻体精准锁定问题本质：不是链路断了，而是出现了两个都自以为唯一的意志。语域上属灾难词汇，事故复盘里说出「脑裂」二字自带警报音效。它与心跳、探活、保活、僵尸进程同属分布式系统的「生理学隐喻体系」——整个学科的词汇集体向人体隐喻靠拢，是系统性隐喻（systematic metaphor）的现成教案：机器的社会，被我们用身体的语言治理。

关　联　主从／主备、心跳、强一致性

墓碑

`tombstone` · *mù bēi*

字　面　坟前刻着死者名字的石碑。

释　义　分布式存储的删除标记：删除不真删，而是写入一条「此键已死」的特殊记录，随复制传播到各副本，待压缩（compaction）时才真正清除。Cassandra、HBase 等 LSM 系存储的标配。

例　句　「这张表天天批量删数据，墓碑堆成山，读一次要跨过一堆墓碑，越来越慢。」

> **语言学注**
>
> 仿译 tombstone，死亡隐喻家族（杀进程、僵尸进程、孤儿进程）里喻体选得最精准的一个：墓碑不是尸体也不是坟，而是「宣告死亡的公开标记」——数据已死，但死讯本身必须作为一条记录存在并广播，否则没收到讣告的副本会让它「复活」。「删除反而要写入」这个反直觉设计，全靠隐喻自身把道理讲圆——此时隐喻不只是命名，而是最好的教学工具。「墓碑堆积」「扫墓碑」等搭配把喻体用到极致，一次慢查询排查读起来像一篇聊斋。

关　联　最终一致性、主从／主备、落盘

写时复制

`copy-on-write（COW）` · *xiě shí fù zhì*

字　面　写的时候才复制。

释　义　多方共享同一份数据，谁要修改谁才真正复制一份出来改，读者继续看旧版。`fork` 的内存页、Redis 的 RDB 快照、各类不可变集合都靠它省内存、免读锁。

例　　句　「bgsave 靠 fork 的写时复制做快照，高写入期内存可能瞬间翻倍，容量得预留出来。」

> **语言学注**
>
> 仿译 copy-on-write：on 表条件（一写就复制），中文以「时」承接，把条件从句压成「X 时 Y」四字格——与「写扩散（写时扩散之省）」同一模板，但这里「时」保住了，因为「写复制」会歧义崩坏。这提示紧缩造词有一条底线：省到歧义为止，能省的只有圈内可脑补的部分。语义上它给一种「拖延哲学」命了名：能不做就不做，拖到不得不做。工程师口语常直接码切为 COW（「这块内存是 COW 的」），字母词与全译自由切换，是语码转换的日常切片。

关　　联　快照隔离、双写、读写锁

第七辑 · 安全与运维

把机器当阵地来守——堡垒、哨兵、蜜罐与金丝雀的军营。

堡垒机

bastion host · *bǎo lěi jī*

字　面　「堡垒」是军事筑城术语，指城防体系中最坚固、突出在外、专挨炮火的据点；「机」缩略自「主机」。

释　义　所有运维登录（SSH/RDP）必须经由的统一入口机，集中做认证、授权、审计与操作录像；国内语境下几乎特指「运维审计系统」这一产品品类。

例　句　「生产环境一律走堡垒机登录，命令有留痕、会话有录像，等保测评就查这个。」

> **语言学注**
>
> 对 bastion host 的逐字仿译（bastion→堡垒，host→机）。原文的 bastion 是棱堡——故意暴露在火线上、牺牲自己撑住全局的突出部，「加固后顶在最前面」才是本义；汉语「堡垒」只剩「坚固」，锐度丢了一半。更有趣的是落地后的语义再窄化：英文 bastion host 泛指任何加固的暴露主机，中文「堡垒机」却在等保合规推动下凝固成一类审计网关产品——译词被本土产业重新定义。新手常按字面把它当防御工事，其实它更像海关关卡：不替你挡子弹，只管验证件、录像存档。

关　联　跳板机、最小权限、零信任、巡检

跳板机

jump server / jump host · *tiào bǎn jī*

字　面　「跳板」本指船与岸之间供人过渡的木板，汉语早已引申为「借以到达更高更远处的中介」（如「出国跳板」）。

释　义　进入隔离网段的中转登录机：先登上它，再从它「跳」进目标内网；与堡垒机的差别在于它只做中转，不必带审计。

例　句　「测试网段不通外网，你先 ssh 到跳板机，再从那儿跳进去。」

语言学注

与英文 jump host 恰好共享「跳」的意象，而「跳板」在汉语里早有现成的中介隐喻，这个仿译等于搭上了本土惯用语的便车，落地毫无异物感。值得记录的是语域反差：黑客语境里「跳板」指用来隐匿来源、横向移动的傀儡机（「拿肉鸡当跳板」），带违法色彩；运维语境里则是堂堂正正的基础设施——同一个词，攻守两侧感情色彩相反，全靠上下文站队。新手最常见的困惑是它与堡垒机的分工，可记作：跳板机管「到得了」，堡垒机管「说得清」。

关　联　堡垒机、提权、红蓝对抗

蜜罐

`honeypot` · *mì guàn*

字　面　装蜂蜜的罐子；喻体来自「甜食诱捕」——熊与蚂蚁循香而至，一头栽进去。英文另有间谍行话渊源（honey trap，美人计式诱捕）。

释　义　故意布设的假系统或假数据，看似有价值且防守松懈，实为诱饵：谁碰它，谁就暴露了自己，一切动作被记录分析。

例　句　「内网那台 3389 大开的『财务服务器』其实是蜜罐，谁去连它，告警直接打到安全组。」

语言学注

教科书级仿译：honey→蜜、pot→罐，逐语素对应，意象无损。妙在语义撞车：汉语本有「蜜罐里泡大的」，喻娇生惯养，甜得纯良；安全行话的「蜜罐」却是杀机四伏的陷阱——同一个词，家常义与对抗义感情色彩完全颠倒，是语域反差的标本。技术谱系可溯至 Stoll《杜鹃蛋》（1989）用假文件钓黑客，到 1999 年 Honeynet 计划成为正式方法论；中文译名随译著与安全社区一同进入，未见音译竞争。理解门槛不在词形而在方向：新手以为它是防御工事，其实它是情报装置——不求挡住谁，只求看清谁。

关　联　哨兵、影子、后门、红蓝对抗

沙箱／沙盒

`sandbox` · *shā xiāng*

字　面　儿童玩沙的围栏箱：在里面随便挖掘、堆砌、推倒，弄不脏外面的地板。

释　义　受控隔离的执行环境：不可信代码在其中运行，对文件、网络、系统调用的访问被严格圈禁；突破隔离叫「沙箱逃逸」。

例　句　「不明附件别直接双击，先扔进沙箱跑一遍，看它有没有偷偷外连。」

语言学注

意译的胜利——没人尝试音译。真正值得记录的是双译并存：安全界说「沙箱」（浏览器沙箱、沙箱逃逸），游戏界说「沙盒」（沙盒游戏），一字之差按行业分流，是罕见的「同源异形、按域分工」案例。另有近音陷阱：军事和应急演练的「沙盘」（沙盘推演，tabletop exercise 的常译）与沙箱毫无关系，可安全会议上三个词常同场出现，新手极易搅浑。隐喻本身对中国读者还打了折扣：sandbox 是美式童年的后院设施，汉语文化里儿童沙池并非默认记忆，许多中文开发者是先学术语、再反推出玩沙的本义——喻体倒灌。

关　联　蜜罐、越权、影子

哨兵

`sentinel` · *shào bīng*

字　面　站岗放哨的士兵：立于边界，警戒异动，敌至则鸣枪示警。

释　义　一指监护进程——盯着主节点、发现失联即触发主从切换（Redis 哨兵模式）；二指算法里的哨兵值/哨兵节点——放在边界上的特殊标记，省去循环里的越界判断。

例　句　「Redis 主库宕了，哨兵自动把从库提成主，业务只抖了两秒。」「链表加个哨兵头结点，删除逻辑就不用单独判空了。」

语言学注

sentinel 的仿译，军事隐喻。两个技术义看似无关，共享的比较点是「替大部队站在边界上」：监护哨兵替业务盯故障边界，算法哨兵替循环体守数组边界——主力因此不必每步回头张望。新手的困惑多来自义项跳跃：数据结构课的「哨兵节点」与运维口中的「哨兵进程」撞名，阿里的流控组件又以 Sentinel 为产品名（做的却是熔断限流），三处「哨兵」各站各的岗。反倒是汉语译名把英文里 sentinel value、sentinel node、Sentinel 进程的散装用法统一了，比源语整齐。

关　联　看门狗、心跳、探针、熔断

看门狗

`watchdog` · *kān mén gǒu*

字　面　看家护院之犬：平日不作声，异动即狂吠，来者不善则咬。

释　义　一种「不喂就咬」的倒计时保护装置：程序须周期性重置计时器（喂狗），一旦超时未喂，即判定系统卡死，强制复位或重启。嵌入式的硬件看门狗、systemd 的软件看门狗同理。

例　句　「主循环卡死超过两秒没喂狗，看门狗直接把板子复位了。」

> **语言学注**
>
> watchdog 的仿译，但它在汉语里长出了英文没有的完整动词生态：「喂狗」（重置计时）、「狗咬了」「被狗复位」（超时触发）——喻体一旦落地就自行繁殖，是仿译词二次构词的绝佳标本。英文说 kick the watchdog，汉语选了更居家的「喂」，乡土气与硬实时系统的冷峻形成强烈语域反差，反而极好记。对新手的门槛是黑话已高度紧缩：「今晚设备又被狗咬死了」，没有喻体框架根本无法解码。注意「看」读阴平 kān（看守义），不读 kàn。

关　联　哨兵、心跳、探针、巡检

探针／存活探针／就绪探针

`probe（liveness / readiness probe）` · *tàn zhēn*

字　面　「探」为伸入试测，「针」为细长器具；本是医学与电子测量词——万用表探针、显微镜探针：以最小创口伸进去感知内部状态。

释　义　周期性健康检查。K8s 里存活探针答「还活着吗」，失败则重启容器；就绪探针答「能接客吗」，失败则摘除流量。APM 语境的「探针」则指植入应用内的采集 agent。

例　句　「就绪探针没配好，滚动发布时流量打到还没预热的实例，5xx 刷了一屏。」

> **语言学注**
>
> 意译加语义窄化：一个泛用的仪器名词，在云原生语境收窄为「定时健康检查」这一个动作。「存活／就绪」的译法极见功力——liveness/readiness 是英文临时后缀化造出的名词，中文用一文一白两个双字词稳稳接住（「存活」口语，「就绪」带公文腔）。新手的坑有二：其一，同一个「探针」在 K8s 指外部拨测、在 APM 指进程内插桩，一词横跨两种架构；其二，两种探针的语义差要靠后果记——答错「存活」会被杀掉重启，答错「就绪」只是暂时没客人；混淆两者，是把服务配成「反复自杀」的经典事故源。

关　联　心跳、看门狗、哨兵、灰度发布

影子／影子库／影子流量

`shadow table / shadow traffic` · *yǐng zi*

字　面　影子与本体同形、同步、寸步不离，却无实体、不改变世界——完美的「只跟随，不作为」。

释　义　影子库/影子表：全链路压测时承接测试写流量的平行表，防止污染真实数据；影子流量：把生产流量复制一份喂给新版本，只记录其反应、不返回给用户。

例　句　「新风控模型先吃两周影子流量，只记结果不拦人，指标对齐了再放开。」

> **语言学注**
>
> 隐喻的比较点选得极准：同形（结构一致）、同步（实时）、无实效（不影响用户）——三点恰是压测与验证所需的全部性质。「影子流量」仿译自 shadow traffic；「影子库/影子表」则大体是国内电商全链路压测实践（双十一体系）里长成的本土用法，属于「隐喻从外语借来、用法在本土长大」。同族的「影子封禁」（shadowban）喻焦点却相反：被遮进影子里的是处罚本身——受罚者看不见。一词多喻、方向不定，新手须逐个学。它与蜜罐构成有趣对照：蜜罐是「假的装成真的」骗攻击者，影子是「真的复制一份」考新系统。

关　联　灰度发布、金丝雀发布、脱敏、蜜罐

蓝绿部署

`blue-green deployment` · *lán lǜ bù shǔ*

字　面　两套对等环境，各涂一色曰「蓝」曰「绿」；一套在线服役，另一套整装待命。

释　义　新版本完整部署到闲置的那套环境，验证、预热完毕后在网关处一刀切换全部流量；出事则原样切回——切换和回退都是秒级的整体动作。

例　句　「这次改动大，走蓝绿：绿环境全量起好、预热完，网关一刀切过去，不对劲秒切回蓝。」

> **语言学注**
>
> 仿译自 blue-green deployment。命名逻辑与金丝雀、灰度截然不同：它以基础设施拓扑命名，而颜色本身刻意无义——相传当年弃用「A/B」，正是怕人误以为 A 是主、B 是备；蓝与绿地位对等、无褒贬、无顺序，这份「故意的任意性」恰是设计思想本身（两套环境轮流为主）。新手问「哪套是蓝的」，标准答案是「哪套都行」——发现问题问错了，概念也就通了。它描述的是非黑即白的瞬时切换，没有中间态；需要中间态，就轮到灰度与金丝雀出场（三者命名逻辑之辨详见「灰度发布」条）。

关　联　金丝雀发布、灰度发布、回滚

金丝雀发布

canary release · *jīn sī què fā bù*

字　面　金丝雀是英国矿工的下井伴侣：这种鸟对瓦斯远比人敏感，鸟先晕倒，人即撤离——用一条廉价而敏感的命，换全队的预警。此俗直到 1986 年才被电子检测仪取代。

释　义　新版本先只承接极小比例流量（如 1%），密切观察错误率与时延；金丝雀健康，再逐步放量，反之立即回滚——用最小的受灾面积探路。

例　句　「先放 1% 流量到金丝雀实例，盯半小时错误率和 P99，没毛病再逐步放量。」

> **语言学注**
>
> 源自英语习语 canary in the coal mine（矿井里的金丝雀＝预警信号），经持续交付文献进入软件业，中文全盘仿译。命名逻辑聚焦「牺牲性探测器」：被切到金丝雀实例的那 1% 用户就是那只鸟——术语自带一点绞刑架幽默。难懂之处在喻体错位：汉语文化里「金丝雀」的默认联想是笼中娇养（乃至「被包养」的俗喻），与「井下哨兵」南辕北辙；不知采矿典故的新手按本土意象解码，只会越想越糊涂——仿译能把习语连根拔来，却拔不来源文化的土壤。同一喻体在安全领域另有分身：栈金丝雀（stack canary），在返回地址前放一个「先死给你看」的哨兵值。

关　联　灰度发布、蓝绿部署、回滚、探针

灰度发布

gradual / phased rollout；所谓 **gray release** 多为中文回译 · *huī dù fā bù*

字　面　「灰度」来自图像技术，指黑与白之间的连续过渡带；新旧版本共存的中间态，就是这段灰。

释　义　按比例、按人群、按地域逐步放量的发布方式：5%→20%→50%→全量，每一档观察数据再前进；也泛指一切「部分用户先见到」的放量机制（灰度开关、灰度名单）。

例　句　「这个功能今晚开始灰度，先放 5% 白名单用户，周五看数据决定要不要全量。」

语言学注

本册少见的本土原创术语：英文世界惯说 gradual rollout 或 canary，「gray release」基本是中文文档的回译品。「灰度」由图像术语隐喻引申并窄化为「发布过程的连续中间态」，在国内互联网工程实践与管理话语（任正非论「管理的灰度」等）相互加持下走红，衍生出灰度环境、灰度用户、放灰、全量等一整套词族。三种放量策略的命名逻辑正好三分：蓝绿部署以环境拓扑命名，状态离散，切换是 0 或 100 的瞬间动作；金丝雀发布以牺牲性探测物命名，焦点在「第一小撮受试者」的预警价值；灰度发布以过程本身命名，焦点在 0 到 100 之间那条连续渐变带——一个指基础设施，一个指受试对象，一个指时间进程。国内口语常拿「灰度」当总称，把金丝雀视为灰度的第一档，语义上确实兼容。

关　联　金丝雀发布、蓝绿部署、影子、回滚

最小权限

`principle of least privilege（PoLP）` · *zuì xiǎo quán xiàn*

字　面　只给最小限度的权限：完成本职所需之外，一概不给。

释　义　安全设计的首要原则（Saltzer 与 Schroeder，1975）：任何用户、进程、密钥只应持有完成当前任务所必需的权限，多一分都是未来事故的引信。

例　句　「新来的实习生直接给 root？按最小权限来，先只开他要用的那两个库的只读。」

语言学注

意译，且是典型的四字格：「最小＋权限」压缩成口号体。汉语技术社区酷爱把原则压成四字——最小权限、纵深防御、按需授权、单一职责——四字格自带成语的韵律与权威感，一说出口就像祖训；这是汉语借传统韵文格式给现代工程原则「加冕」的社会语言学现象。代价是紧缩丢掉了原文的 principle 与「least」的比较级语感，新手容易当成某个具体配置项（「最小权限在哪里打开？」），而它其实是一种设计伦理。日常动词化说法是「权限收一收」「按最小化给」。

关　联　零信任、提权、越权、堡垒机

纵深防御

`defense in depth` · *zòng shēn fáng yù*

字　面　「纵深」是军事术语：阵地沿敌进攻轴线向后延伸的深度。不指望一线挡住敌人，而是层层设防、逐层消耗。

释　义　不依赖任何单一防线：WAF、网络分段、主机加固、密钥轮换、审计告警层层叠加，任何一层被穿透，后面还有下一层兜着。

例　句　「别指望 WAF 一层挡光注入，纵深防御：参数校验、预编译、库账号最小权限，一层破了还有下一层。」

> **语言学注**
>
> 这条的翻译路径罕见地「军味无损」：defense in depth 本是正牌军事学说（一战弹性防御以降），进入信息安全后再译入中文；而「纵深」早在上世纪五十年代就随苏军大纵深战役理论的译介成为汉语军语——安全术语落地时，现成的军事词库严丝合缝地接住了它，堪称「军语对军语」的无缝仿译，也解释了它在中文里为何毫无翻译腔。四字并列格更添操典气质。新手的障碍在「纵深」本身已是生僻军语：不明白「深」指空间上的纵向冗余，常误解为「深入的防御＝防得很用心」。与零信任是两代哲学：纵深防御承认边界终将被突破，零信任干脆否认边界的意义。

关　联　零信任、最小权限、堡垒机、蜜罐

零信任

`zero trust` · *líng xìn rèn*

字　面　信任为零：不因你身处内网、持有工牌就默认放行；每一次访问都重新验明正身。

释　义　一种安全架构（Forrester 分析师 Kindervag 于 2010 年命名，Google 的 BeyondCorp 实践使其落地成名）：取消「内网即可信」的默认假设，所有访问不论内外，一律持续认证、按最小权限授权。

例　句　「都上零信任了，内网调用照样要 mTLS 加设备认证，别再迷信『进了内网就是自己人』。」

> **语言学注**
>
> 仿译，且带来了构词层面的借用：「零X」原非汉语能产格式（旧说「无X」「不X」），随 zero tolerance→零容忍、zero-day→零日、zero-copy→零拷贝一路复制，「零」已升格为类前缀——英语构词模板整体移植入汉的标本，零信任是其中最风行的一例。语义上它是营销式的夸张缩略：并非「什么都不信」（那系统没法用），而是「不存在基于网络位置的隐式信任」；新手按字面较真会立刻抬杠「那还怎么干活」，须把「零」读作「信任默认值为零、逐次挣得」。与纵深防御对照记忆最佳：一个层层设防，一个釜底抽薪。

关　联　纵深防御、最小权限、堡垒机、越权

加盐

salt / salting · *jiā yán*

字　面　做菜撒盐：同一道菜，盐量不同则味道不同。给每个密码哈希拌进一撮随机数据，让相同的密码也算出不同的哈希。

释　义　存储口令时为每个用户生成随机盐值，与口令拼接后再哈希，使彩虹表（预计算的哈希反查表）失效；盐可明文存放，机密的那份叫 pepper（胡椒），行话罕译。

例　句　「用户表的哈希没加盐，一张彩虹表全反出来了，赶紧换 bcrypt 并强制重置口令。」

> **语言学注**
>
> 动宾紧缩的典范：salt 名词进入汉语即配上「加」动词化，两字成词，会话中可直接作谓语（「密码加没加盐？」）。厨房语域嵌进安全评审的严肃语境，反差强烈却过耳不忘——生活隐喻是最好的记忆钩子。术语史可溯至 1970 年代 Unix crypt 的 12 位盐；中文社区从未音译。新手最常见的误解恰好藏在喻体里：以为盐像秘方须保密，实则盐防的是「批量查表」而非「被人知道」，明文存在哈希旁边毫无问题——喻体只保证「每道菜味道不同」，不承诺「配方无人知晓」。同灶的「胡椒」倒真要锁进保险柜，中文口语多直接说 pepper。

关　联　撞库、脱敏、最小权限

脱敏

data masking；中式英文常作 **desensitization** · *tuō mǐn*

字　面　脱去敏感性。本是医学词：脱敏治疗，让过敏体质对花粉尘螨不再剧烈反应；牙膏也有「脱敏」款。

释　义　把数据里的隐私字段变得不可识别再流出生产环境：手机号打星、身份证遮位、姓名换假名；分静态脱敏（导出前批量处理）与动态脱敏（查询时实时遮盖）。

例　句　「导给外包联调的库必须先脱敏，手机号中间四位打星，身份证留头尾。」

> **语言学注**
>
> 不是外来译词，而是本土医学术语的跨域挪用：「敏感数据」的「敏」接上现成的动宾结构「脱敏」，浑然天成。反倒是英文没有对应惯用语——英语圈说 data masking、redaction、anonymization，「data desensitization」主要见于中国公司的英文文档，是罕见的汉→英回译，可作术语输出的小案例。两字动宾在合规话语里效率极高：「脱过敏了吗」四字完成一次审计问询。新手易把脱敏、加密、匿名化混为一谈：加密可逆（有钥匙就能还原），脱敏通常不可逆但保留格式，匿名化则要求连关联推断都切断——三者合规含义差别巨大，拿脱敏当匿名化用是数据出境审查里的经典翻车点。

关　联　加盐、影子、最小权限、撞库

越权

`broken access control / unauthorized access` · *yuè quán*

字　面　越过权限边界。「越」的语素家族一脉相承：越界、越狱、僭越——从礼制到国境到牢笼，皆是跨过不该跨的线。

释　义　程序没管住「谁能动谁的数据」：水平越权指同级用户互访（改个 orderId 就看到别人的订单），垂直越权指低权限者做了高权限操作。OWASP 榜单上多年的头号漏洞类别。

例　句　「把 URL 里的 orderId 改成别人的也能看，典型的水平越权，上线前必须修掉。」

> **语言学注**
>
> 动宾紧缩，但词根古老：「越权」原是行政与法律语汇（越权行政、越权代理），计算机安全将其语义窄化为访问控制缺陷这一具体门类，并发展出「水平/垂直」的二维分类——英文得用 IDOR、privilege escalation 等好几个词覆盖的范围，中文一个词根加两个方位修饰就收编了，分类反而比源语整齐，是译入语后来居上的例子。更古的祖先是「僭越」：孔子怒斥「八佾舞于庭」，本质上就是垂直越权——礼制即最早的访问控制。与「提权」的分工见彼条：越权是「用了不该用的」，提权是「变成了不该是的」。

关　联　提权、最小权限、零信任

提权

`privilege escalation（privesc）` · *tí quán*

字　面　提升权限：从低权限账户爬到 root 或 SYSTEM。

释　义　攻击链中段的关键一跳：先以低权限落脚（如 Web 进程），再利用内核漏洞、错误配置或凭据搜刮升为管理员；「本地提权」（LPE）是漏洞公告的常客。

例　句　「拿到的只是 www 用户的 shell，内核挺老，试试脏牛提权。」

> **语言学注**
>
> 又见双字动宾紧缩，且是攻击方视角的行话：红队说「提权」，官方公告写「权限提升漏洞」——同一概念的松紧两式按语域分工。这类双字动宾是中文安全黑话的主力构词模板：提权、越权、撞库、拖库、爆破、抓包、挂马……短促及物，念起来像作战口令，把复杂攻击动作压成两拍，是行话追求经济性与身份标记的双重产物。「提」与「越」二字精确编码了两种几何：提权是纵向爬升（改变身份等级），越权是横向跨界（身份不变而手伸过界）——新手混用二词，老手一听便知深浅。

关　联　越权、最小权限、后门、红蓝对抗

巡检

routine inspection；无地道英文对应词　·　*xún jiǎn*

字　面　「巡」为按路线走动查看（巡逻、巡视），「检」为查验。两个近义语素并列，自带「沿固定路线逐点查看」的画面。

释　义　周期性主动检查系统健康状况：磁盘水位、证书有效期、备份任务、慢查询、告警静默项，按清单逐项过一遍；如今多写成巡检脚本或平台化。

例　句　「大促前一周，每天早晚各一轮巡检：磁盘、证书、备份、慢 SQL，全过一遍打钩。」

> **语言学注**
>
> 本册的「土著词」代表：它不是任何英文词的翻译——英语只有 routine check、health check 等松散说法，没有一个词能带出「沿线走一圈」的巡逻意象。它从电力、铁路、机房等传统工业直接继承（电力巡检工沿杆塔走线是字面意义的「巡」），把蓝领的勤勉气质一并带进互联网运维；语体上比 monitoring 多一层「人到、眼到、签到」的责任伦理。这也解释了自动化时代它为何不死：说「自动化巡检」而不说「多配点监控」，强调的正是逐项打钩的仪式感。新手（尤其海归）会因找不到英文对译而困惑——答案是：没有，这词是我们自己的。

关　联　值班、探针、告警风暴、看门狗

值班／on-call

`on-call` · *zhí bān*

字　面　「值」为轮到、当值（旧衙门与军营皆有当值、值宿之制），「班」为班次：轮到你的班，你就守着。

释　义　按排班表承担非工作时间的故障响应义务：告警来了要接、要处置、要升级。互联网公司口语常直接说 on-call。

例　句　「我这周 on-call，凌晨三点被告警电话叫起来，处理完顺手把值班手册补了两条。」

> **语言学注**
>
> 本土古词与外来字母词同场竞争的活标本。「值班」在汉语里有几百年基业，但语义默认「人在岗位上」；on-call 的要义恰是「人可以不在，但必须叫得应」——一个守「位」，一个守「联」。这道语义缝隙给了字母词立足之地：说「这周我 on-call」精确传达「带着电话过日子」的状态，说「我值班」倒像要去机房坐着。于是出现按企业文化分流的词汇社会学：电信、银行、国企说值班、值守、7×24；互联网大厂说 on-call、oncall 群、排班轮换——选词即站队，一听便知你在哪种公司上班。这是字母词填补语义空位、而非单纯崇洋的好例。

关　联　告警风暴、巡检、复盘、根因分析

告警风暴

`alert storm / alarm flood` · *gào jǐng fēng bào*

字　面　告警如风暴：一瞬间铺天盖地，能见度归零。

释　义　一个根因沿依赖链放大成成百上千条告警（核心交换机闪断，下游所有服务同时报超时），监控群刷屏，真正的病灶反被噪声淹没。治理手段叫告警收敛、告警降噪。

例　句　「核心交换机闪断 30 秒，监控群瞬间告警风暴，几百条一起刷，真正的根因反而被淹了。」

语言学注

气象隐喻的仿译（alert storm），概念谱系却来自流程工业与航空的报警管理研究——控制室报警泛滥致人麻木酿成事故，早有成文标准；IT 监控把这套人因工程词汇整体搬来（告警疲劳＝alarm fatigue 同源）。汉语内部另有一道值得记录的语用分工：「告警」与「报警」——日常汉语「报警」默认是打 110，运维圈遂偏用「告警」指机器发出的 alert，一字之差划开「人报案」与「机示警」两个世界；新手文档里写「系统报警了」，老手都会顺手改成「告警」。治理词也各有喻源：「收敛」借自数学，「降噪」借自音频——一个术语生态里，三种学科的隐喻在协同工作。

关　联　值班、根因分析、巡检、复盘

根因分析／根因／RCA

`root cause analysis（RCA）` · *gēn yīn fēn xī*

字　面　root→根、cause→因，逐语素仿译后再紧缩：根本原因→根因。树根隐喻：症状是枝叶，病根在土里，不刨到根，砍多少枝都会复发。

释　义　事故处置的下半场：沿时间线层层追问（五问法、鱼骨图皆是其工具），找到那个「修掉它，这类事故就不再发生」的原因，并落成改进项。口语常直接用字母词：「出个 RCA」。

例　句　「这次故障的 RCA 明天出初稿：时间线、根因、影响面、改进项，一条条列清楚。」

语言学注

三种机制叠放在一个词条里：先仿译（root cause→根本原因），再双音节紧缩（→根因，服从汉语双字词的韵律强迫症），会话里又常整个让位给字母词 RCA——书面「根因分析」、口头「RCA」，语码转换按媒介分层的典型。概念史来自制造业质量管理（丰田五问、石川鱼骨图）与核电航空的事故调查，经 SRE 文化进入互联网。「根因」二字对新手的暗坑是单数错觉：树根隐喻诱导人找「那一个」根，而复杂系统的事故几乎总是多因叠加，成熟的复盘文化因此强调「贡献因素」而非唯一元凶——隐喻塑造思维，也会框限思维。

关　联　复盘、告警风暴、回滚、值班

撞库

`credential stuffing` · *zhuàng kù*

字　面　拿着从别处泄露的账号密码，一条条往目标网站的登录口「撞」——像捡了一串钥匙挨家试锁，撞开哪家算哪家。「库」即数据库。

释　义　利用用户到处复用密码的习惯，把 A 站泄露的凭据批量试到 B 站；防御靠加盐慢哈希、异常登录风控与多因素认证。

例　句　「某电商泄的那批账号密码正被拿来撞库，登录接口异地失败率飙升，先上验证码兜底。」

> **语言学注**
>
> 本册最值得记录的本土原创黑话：它不是 credential stuffing 的翻译——英文喻体是往表单里「塞填」凭据，汉语喻体是「碰撞＋碰运气」（撞大运的「撞」），两边各自造词，中文这个更传神。它属于一条完整的黑产动词流水线：拖库（把整库数据拉走；早期论坛谐音戏写作「脱裤」，库被扒走如裤子被扒，粗俗双关正是地下行话的身份暗号）→洗库（榨取变现）→撞库（拿到别家复用）——三个双字动宾串成产业链叙事，构词效率与画面感俱佳，也标记着 2011 年前后国内多起大型明文密码泄露事故催生的行话年代层。新手看字面猜不出「撞」的是登录接口——须补上「密码复用」这块拼图，词义才咬合。

关　联　加盐、脱敏、后门、红蓝对抗

后门

`backdoor` · *hòu mén*

字　面　建筑后门：正门有岗哨门禁，后门通厨房小巷——绕过检查的隐蔽通道。

释　义　绕过正常认证的隐蔽访问通道：可能是开发者留的调试入口忘了删，可能是攻击者植入的持久化访问，也可能是供应链投毒埋进依赖里的暗桩。

例　句　「审计发现网关上留了个免鉴权的调试后门，直连内网，连夜下掉并全量换了密钥。」

> **语言学注**
>
> backdoor 的仿译，却落在一个早已被占用的汉语词位上：「走后门」在计划经济年代就是托关系绕规则的国民俗语。两义共享同一语义内核——绕开正门的合法检查——所以这次借译毫无违和，甚至互相增味：给程序「留后门」听起来天然带着「走后门」式的道德阴影。动词搭配划分立场：留后门（内部人的预谋）、开后门（放行）、植入后门（外部攻击）——「留」字最诛心，一个字写尽监守自盗的故意。概念史上有 Thompson 1984 年图灵奖演说里自我繁殖的编译器后门作镇册之宝，近年 XZ 供应链事件又让它重回头条。新手要防的反而是词太日常带来的钝感：安全语境里它是最高严重级，不是俗语里那点小猫腻。

关　联　蜜罐、提权、撞库、零信任

红蓝对抗／红队／蓝队

red team vs. blue team · *hóng lán duì kàng*

字　面　红队与蓝队捉对演习。颜色码源自军事推演：普鲁士兵棋以蓝色代表己方（普军制服为蓝），敌方涂红；冷战后西方沿用「红方＝威胁」。

释　义　以真实攻击手法检验防御体系的演练：一方扮攻击者渗透突防，一方值守监测响应；国内每年的护网行动（HVV）是最大规模的行业化红蓝对抗。

例　句　「下个月护网，我们先内部来一轮红蓝对抗摸底；方案里写明这次红队是攻击方，免得开会鸡同鸭讲。」

> **语言学注**
>
> 仿译撞上颜色语义的文化对冲，是本册最佳的符号学案例。西方谱系里 red team＝攻击方（红色自冷战起编码为威胁）；解放军演习传统恰相反：红军是自己人（红色是革命自我认同），蓝军才是假想敌（专业陪练如朱日和蓝军旅）。两套颜色码随各自文献一起涌入国内安全行业，谁攻谁守全看说话人师承，以致合同和演练方案里常须明写「本次红队为攻击方」来自我消歧——一个颜色词在两种政治文化里指向相反的敌我，术语翻译踩进了符号学深水区。衍生词继续玩颜色：紫队（红蓝协同复盘，红加蓝调成紫）、白帽黑帽（西部片正邪帽色经黑客文化转手而来）。

关　联　蜜罐、提权、撞库、纵深防御

复盘

postmortem / after-action review · *fù pán*

字　面　围棋术语：对局结束后把棋子按原顺序重摆一遍，逐手推敲得失。「盘」即棋盘。

释　义　事故平息后的结构化回顾：拉时间线、定根因、算影响面、列改进项；成熟团队强调「对事不对人」（blameless）。

例　句　「周四故障复盘会，时间线我来整理，大家提前把各自环节的日志截好图，对事不对人。」

语言学注

与英文 postmortem 对照是一堂比较隐喻学课：英语选了法医意象——事故是一具尸体，剖开查死因，冷峻而自带追责气味，以致须专门加 blameless 消毒；汉语从棋道取喻——事故是一盘输棋，重摆一遍为的是下一盘赢，学习性与建设性内置于词根，「无责」几乎不必另行声明。这个围棋古词先被联想等企业的管理话语发扬成方法论，再顺着互联网公司流进工程师的事故文化，恰好填上 postmortem 的词位——不是翻译，是本土词对进口概念的「认领」。它与根因分析分工明确：复盘是仪式与叙事（会要开起来），RCA 是产出物与交付件（文档要写出来）。新手当心语义膨胀：如今连约会失败都能「复盘」，工程语境里请守住「有时间线、有改进项」的硬核用法。

关　　联　根因分析、值班、告警风暴、回滚

第八辑 · AI 与大模型时代

最新一批行话，把化学、心理与炼丹一起搬进了机器。

语料

corpus · *yǔ liào*

字　面　语言的「材料」。「料」本指待加工的原料（木料、布料、饲料），把语言当成了可称重、可加工的物资。

释　义　用于训练或评测模型的成规模文本（或语音、代码）集合；正式全称「语料库」是 corpus 的经典译名。

例　句　「中文语料里论坛体占比太高，模型一开口一股贴吧味，得再配比些书面语料进去。」

> **语言学注**
>
> 意译加隐喻引申。语言学界译 corpus（拉丁语本义「身体」）为「语料库」已数十年，大模型时代把「库」字磨掉，将「语料」当不可数名词论吨称量：「喂了 2T 语料」。「料」这个语素把语言彻底物化为生产资料，于是「清洗」「去重」「配比」等加工动词全部顺理成章——一整套工厂话语，暴露出这个行业对语言的原材料态度。另注意它比「数据」窄：专指语言类数据，初学者常混用二者。

关　联　预训练、喂数据、词元、分词

词元／令牌／token

token · *cí yuán / lìng pái*

字　面　「词元」＝构成词句的最小单元，「元」取「单元、元件」义；「令牌」＝旧时出入关防的信物腰牌，喻体是身份凭证。

释　义　模型处理文本的最小切分单位（可能是词、子词或几个字节），也是上下文长度与计费的计量单位。

例　句　「光系统提示词就占了两千多 token，上下文窗口再大也经不起这么烧。」

> **语言学注**
>
> 译名「未定于一尊」的活标本。英文 token 一词多义，中文各分支各译各的：编译原理译「记号／词法单元」，操作系统与安全译「令牌」（令牌环、访问令牌，取凭证义），NLP 新铸「词元」（取最小单位义）。LLM 走红时「令牌」已被安全领域占定，教材与术语审定遂倾向「词元」；可口语几乎全用字母词——「token 烧得快」「按 token 计费」——因为它天天出现在 `max_tokens` 这类 API 字段和账单里，字母词与代码零转换对齐，任何汉译反而多一道心译工序。书面「词元」、口语「token」的分层并存，正是术语定名竞争的进行时。

关　联　分词、上下文窗口、语料

微调

`fine-tune / fine-tuning` · *wēi tiáo*

字　面　轻微地调节。喻体是旋钮：老收音机面板上就有「微调」钮，英文 fine-tune 同样出自调校收音机与乐器的场景。

释　义　在预训练模型基础上，用小规模领域或任务数据继续训练使其适配特定用途；细类有指令微调（SFT）、LoRA 微调等。

例　句　「通用模型对机械制图术语一知半解，拿三千条标注问答微调一版试试。」

> **语言学注**
>
> 状中式紧缩（「微」修饰「调」），更难得的是一次「现成对译」：中英两边各自早用旋钮隐喻造好了这个词（「政策微调」久已通行），AI 时代只需握手，翻译损耗近乎为零，竞争译名「精调」被挤到论文角落。语义窄化倒很明显：日常「微调」指举手之劳的小改，AI 的「微调」却是一整套可能烧卡数日的训练流程——「微」描述的是相对预训练的量级，不是绝对工作量，初学者常被这个字骗得低估成本。

关　联　预训练、蒸馏、对齐、炼丹

蒸馏

`knowledge distillation` · *zhēng liú*

字　面　加热液体使汽化再冷凝，以提纯分离。喻体是化学实验室的蒸馏瓶。

释　义　让小模型学习大模型的输出分布从而继承其能力（知识蒸馏）；日常泛指「用大模型的输出训练小模型」。

例　句　「端侧那版 7B 是从 70B 蒸出来的，指标掉了两个点，推理成本只剩十分之一。」

> **语言学注**
>
> 隐喻引申里的「二手进口」：Hinton 先在英文里把化学的 distillation 搬进机器学习，中文不必新译，直接续用化学课本里现成的「蒸馏」——隐喻连同译名一起打包到岸，这是「蒸馏、温度、涌现、收敛」这族理科隐喻的共同来路。喻体映射也格外工整：蒸馏取精去粗、产物更小更纯，正对「能力浓缩、参数变少」。圈内已把它动词化拆用——「蒸出来的小模型」「拿 70B 蒸个 7B」——单字「蒸」即可指代全流程，说明隐喻已完全词汇化。其褒义预设值得警惕：「提纯」的想象掩盖了蒸馏必有损耗的现实。

关　联　量化、微调、大模型／小模型

量化

`quantization` · *liàng huà*

字　面　把连续量切成离散档位。喻体来自物理与信号处理：模数转换把连续波形切成台阶，「量」与量子（quantum）同源。

释　义　把模型权重从高精度浮点压到低位宽（如 FP16 到 INT4）以省显存、提速度，代价是精度损失（「掉点」）。

例　句　「这张卡 24G 显存跑不动全精度，先量化到 4bit，掉点能接受就上。」

> **语言学注**
>
> 一词双源的干扰样本。汉语里「量化」早被社会科学与金融占用：量化考核、量化宽松、量化交易，取「把定性的变成可度量的」之义，是「-化」缀使动构词；而 AI 的「量化」承自信号处理对 quantization 的旧译，专指压低数值精度。两义共享字面、方向却几乎相反——前者从模糊走向精确，后者牺牲精确换取轻量——初学者望文生义十有八九栽在这里。工程口语里它已带上补语和评价体系：「量化到 4bit」「量化完掉点了」，「掉点」这样的行话替它补足了代价维度。

关　联　蒸馏、端侧／本地部署、大模型／小模型

幻觉

`hallucination` · *huàn jué*

字　面　无中生有的感知。喻体来自精神病学：没有外界刺激却「看见」「听见」不存在的东西。

释　义　模型以流畅笃定的口吻输出与事实或给定材料不符的内容，如编造 API、虚构文献出处。

例　句　「它把一个不存在的函数签名答得有鼻子有眼，这种幻觉比直接报错难防多了。」

> **语言学注**
>
> 仿译叠加拟人化的感情色彩工程。术语借自精神病学，把统计性错误包装成「病症」：既拟人（暗示模型有感知、会「看错」），又免责（是犯病，不是撒谎），比民间的「一本正经地胡说八道」体面，也比学界另一候选「虚构（confabulation）」传播力强。语义窄化同样彻底：AI 的「幻觉」不涉感知，专指「流畅而失实的生成」，且已中性化为可度量指标（幻觉率）。批评者指出这个词高估了模型的心智、又淡化了厂商的责任——一枚译名，承载一场立场之争。

关　联　对齐、护栏、温度、检索增强

提示词／提示

prompt · *tí shì cí*

字　面　提点、递话头的话。「提示」原用于舞台提词、考试提示；英文 prompt 的喻体正是剧场里的提词员（prompter）。

释　义　发给模型以引出目标输出的输入文本，包括系统指令与用户消息；也泛指调用时组装的整段输入。

例　句　「提示词别硬编码在代码里，抽到配置里去，回头 A/B 两版看效果。」

> **语言学注**
>
> 意译与字母词并轨：书面写「提示词」，口语常直接说 prompt，还长出动词用法「prompt 它一下」，是典型的语码转换。「词」缀是点睛之笔——单说「提示」动词性太强，又与 UI 语境的「提示」（弹窗、气泡）撞车，加「词」才把它定形为一件可编辑、可版本化的工件。只是这个「词」早已名不副实：真实提示词动辄千字，是「文」而非「词」——字面与所指的错位，恰好记录了这个概念几年间的膨胀速度。

关　联　提示工程、上下文窗口、温度

提示工程

prompt engineering · *tí shì gōng chéng*

字　面　把写提示词抬举为一门「工程」。喻体来自土木、软件等正规工程学科。

释　义　系统化地设计、测试、版本化提示词以稳定获得目标输出的方法论：少样本示例、角色设定、输出格式约束等。

例　句　「这不是玄学，是提示工程：把输出钉死成 JSON 格式，再塞两个反例，成功率立刻上来。」

> **语言学注**
>
> 仿译自 prompt engineering，修辞重心全在「工程」二字的语域抬升：把「会跟模型说话」这件看似人人都会的事冠以工程之名，以宣示其可复现、可度量、值得开工资。感情色彩因此两极——从业者用它自证专业，圈外人拿「写作文也算工程师」调侃；它与本土戏称「念咒」构成语域反差：同一件事，一头够向工程学，一头贬作巫术，恰说明其方法论地位未稳。如今该词又被「上下文工程」蚕食，术语自身的快速代谢，就是这个领域节奏的写照。

关　联　提示词、上下文窗口、智能体、思维链

上下文窗口

`context window` · *shàng xià wén chuāng kǒu*

字　面　「上下文」＝某处之前之后的文字；「窗口」＝只露出一截的开口。喻体是滑动窗口：隔着窗框看长卷，看得见的永远只有窗内一段。

释　义　模型单次调用能容纳的最大 token 数，输入输出共享额度，超出即截断或报错。

例　句　「整个 repo 塞进去早爆窗口了，先做检索，只把相关文件片段放进上下文。」

> **语言学注**
>
> 两枚旧译的拼装：「上下文」是计算机圈对 context 的传统译法（上下文切换、进程上下文），语言学里同一概念多译「语境」——AI 选了计算机系谱；且「上下文」字面锚死线性文字，到多模态时代已开始漏风（图像算什么「文」？）。「窗口」则承自信号处理与 TCP 的滑动窗口。四字连用读来俨然成语，实为技术复合词；口语常缩略成「窗口」，配「爆」「塞」「挤」等空间动词——「爆窗口」把抽象的 token 上限体验成容器溢出，是隐喻落地为日常动宾搭配的好例。

关　联　词元、提示词、检索增强、注意力

对齐

`alignment` · *duì qí*

字　面　使两物排齐对准。喻体来自排版与装配车间：对齐边线、对齐孔位。

释　义　使模型行为与人类意图、偏好及价值观一致的目标与技术（RLHF、宪法式 AI 等）；「对齐税」指为此损失的能力。

例　句　「这版模型对齐做得太狠，正常的安全测试问题也一律拒答，用户天天骂。」

> **语言学注**
>
> 语义窄化的极端个案：日常「对齐」是几何动作，AI 义却是「让机器价值观贴合人类」这样宏大的伦理工程——字面越小，所指越大，落差里装着整个领域的雄心与焦虑。措辞还悄悄改写了论元结构：被对齐的是模型，对齐的基准（谁的价值观？）常年隐而不说，「对齐了没有」听着像拧螺丝，实则是无法验收的开放难题——把伦理问题动作化，中文「对齐」比英文 alignment 更轻描淡写。派生的「对齐税」「超级对齐」证明它已是能产词根。另防撞车：大厂黑话「跟业务对齐一下」（开会同步）与此同形异义。

关　联　护栏、越狱、幻觉

涌现

emergence · *yǒng xiàn*

字　面　如泉水涌出、显现。「涌」自带突然、大量、自下而上的水势意象。

释　义　模型规模越过某阈值后，训练目标未直接教过的能力（多步推理、写代码）忽然可用的现象；固定搭配「涌现能力」。

例　句　「30B 以下怎么测都不会的任务，到 70B 忽然就会了——论文叫涌现，我们私下叫玄学。」

> **语言学注**
>
> 理科隐喻「打包进口」的又一例：emergence 先在复杂性科学与心灵哲学安家（整体大于部分之和），中文旧有「突现」「层展」「涌现」诸译并竞，AI 圈选定了最有画面感的「涌现」。这个译名甚至比原词生动：emerge 只是「浮出」，「涌」添了水势，暗示能力从亿万参数底下自行喷薄，无人安排却自己到场——近乎神秘主义的褒义，营销文案最爱；而学界始终有人泼冷水，说所谓涌现不过是评测指标不连续造成的错觉。一个词同时充当科学假说、营销话术与批评靶子，是大模型话语场的缩影。

关　联　大模型／小模型、预训练、注意力

过拟合

overfitting · *guò nǐ hé*

字　面　「拟合」＝用曲线去凑数据点（拟：比拟；合：吻合），「过」＝过度。喻体来自统计制图：曲线硬穿过每个点，扭成麻花。

释　义　模型把训练数据的噪声与偶然规律也背了下来，训练集漂亮、新数据上垮掉；泛指「只记住，没学会」。

例　句　「验证集 loss 都开始回升了还接着训，这不明摆着过拟合了吗？」

> **语言学注**
>
> 词缀级仿译：over- 对「过」、under- 对「欠」，一对反义术语整套移植，系统性罕见。难点在「拟合」这个数学旧译语素不透明，初学者很难从字面还原「曲线凑点」的图景，往往要亲眼见过那张三条曲线的对比图才恍然大悟——这是一个必须靠图像才能激活隐喻的术语。它的语域下移倒很成功：「他做题过拟合了，换个问法就不会」，从术语沦为讥讽应试的俏皮话，说明「死记硬背 vs 举一反三」这层引申早被大众接住。

关　联　欠拟合、泛化、收敛

欠拟合

`underfitting` · *qiàn nǐ hé*

字　面　「欠」＝亏欠、不足：拟合得欠火候，曲线懒得连训练数据都凑不上。

释　义　模型容量或训练不足，连训练集都学不好，处处差一口气。

例　句　「训练集准确率才六成，这是欠拟合，先加容量或多训几轮，别急着怪数据。」

> **语言学注**
>
> 与「过拟合」共享模板，「欠」译 under- 颇见巧思：不取生硬的「不足拟合」，单音节前缀既保住三字格节奏，又接上汉语「欠揍」「欠考虑」这类「欠 X」能产格式的口语底气。更有意思的是一对术语的冷热不均：过拟合人人挂嘴边，欠拟合近乎失语——在「大力出奇迹」的规模竞赛年代，「容量不够」听起来简直是道德缺陷，没人愿意谈。术语使用频率的不对称，折射的是算力军备时代的集体心态。

关　联　过拟合、泛化、收敛

召回率

`recall` · *zhào huí lǜ*

字　面　「召回」＝把散在外面的一一召唤回来，如点名收队；今人更熟的联想是厂商召回问题车。

释　义　所有真实正例中被模型找出来的比例，衡量「漏没漏」；与精确率此消彼长。

例　句　「安全审核宁可误杀不能漏放，召回率优先，精确率靠人工复核兜底。」

语言学注

意译中的歧义规避：recall 在心理学是「回忆」，信息检索沿用此喻——从记忆库里把相关文档「想起来」；中文若直译「回忆率」便滑进心理学语域，译者改「召回」，把主体从「想起」换成「召集」，动作感更强，还顺手蹭上「汽车召回」的日常联想——助记却也误导（此召回与缺陷无关）。图书情报学派更早的译名「查全率」（与「查准率」配对）如今在机器学习语域败下阵来，同一概念两代译名，新陈代谢肉眼可见。真正折磨初学者的是它与「精确率」的镜像纠缠，多数人得靠「宁可错杀（要召回）还是宁可放过（要精确）」的口诀才站得稳。

关　联　精确率、检索增强

精确率

`precision` · *jīng què lǜ*

字　面　精细而准确的比率。「精确」由日常形容词直接升格为度量名。

释　义　模型判为正例的结果中真正例的占比，衡量「报出来的有多少是真的」；与召回率配对使用。

例　句　「模型标了两百处疑似缺陷，人工一核只有一百四是真的，精确率七成，误报还得压。」

语言学注

麻烦不在翻译在近义拥挤：precision（精确率）与 accuracy（准确率）在日常汉语几乎同义，术语上却各有严格定义，「精确不等于准确」这种违背语感的切分是入门第一道坎——英文本就有 precision/accuracy 之辨，中文照单全收，又因「精」「准」「确」三个语素可自由组合而更乱：正确率、查准率各占山头（后者是信息检索旧译）。跨学科再添堵：计量学里 precision 译「精密度」、accuracy 译「准确度」，同一对英文词在两个学科领了两套汉名。术语系统的整齐，在这一族词上败给了历史层积。

关　联　召回率、幻觉、过拟合

收敛

`convergence` · *shōu liǎn*

字　面　收拢、约束（本可用于人：「你收敛点儿」）；数学义为序列无限逼近某定值。喻体是「散的聚到一点」。

释　义　训练中损失趋于平稳、不再明显下降，即「训到头了」；引申指讨论、需求从发散走向定案。

例　句　「loss 三千步之后基本收敛，再训就是烧钱，早停吧。」

> **语言学注**
>
> 数学分析老译名的再就业：converge 译「收敛」由来已久，AI 承其数学义，却把判定现场从黑板挪到 TensorBoard 曲线——标准也从严格的极限定义松弛为「看起来平了」，语义在工程语域里明显软化。它还完成了第二跳：从「训练收敛」外溢为「需求收敛」「方案收敛」，程序员拿它形容会议从发散到拍板，等于用技术腔重新激活了「收敛」的日常义，属于术语反哺通用语。一词三层——数学极限、曲线形状、会议进度——语域漂得越远，含义越松。

关　联　过拟合、欠拟合、炼丹

温度

temperature · *wēn dù*

字　面　冷热程度。喻体直通统计物理：玻尔兹曼分布中温度越高，粒子越躁动、状态越随机。

释　义　控制采样随机性的超参数：低温保守稳定，高温发散多样；`temperature=0` 近似确定性输出。

例　句　「写合规文案把温度压到 0.2，脑暴 slogan 再拉到 1.2，同一个模型两副性格。」

> **语言学注**
>
> 这族理科隐喻里血统最纯的一支：softmax 采样公式直接搬自统计力学，连符号 T 都没换——「温度」不是修辞，是公式里的字面参数，隐喻引申与术语借用在此重合。体感还极准：高温「热」得天马行空，低温「冷」得字斟句酌，中文使用者凭生活经验即可预期其效果，堪称零学习成本的参数名。围绕它的动词搭配「打到 0」「拉高点」延续旋钮意象，与「微调」同属一个操作台隐喻家族。常见误会是拿温度当「准确率旋钮」——它只管随机性，低温照样一本正经地幻觉。

关　联　幻觉、提示词、推理

越狱

jailbreak · *yuè yù*

字　面　翻墙逃出监狱。喻体先经 iPhone「越狱」（破解系统封锁）中转，再抵达 AI。

释　义　用特殊构造的提示词绕过模型安全限制，诱其输出被禁止的内容；相应文本称「越狱提示词」。

例　句　「上周疯传的那段角色扮演越狱已经失效了，官方明显打了补丁。」

> **语言学注**
>
> 仿译的二级传播：jailbreak 从真监狱到 iOS 破解已隐喻化一次，中文照译「越狱」接盘全部语义史，AI 义是第三跳——多数中文用户其实是从 iPhone 而非监狱学会这个词的，喻体在集体记忆里早被替换。它的框架效应值得咂摸：「狱」预设模型本被合法关押、厂商是狱方，攻击者自带劫狱的浪漫，比中性的「绕过安全策略」多出一整套政治想象，用起来常带三分武勇七分戏谑。与「护栏」对读更妙：一边把安全措施想象成高墙铁窗，一边想象成路边栏杆，两个隐喻对模型「自由」的裁定针锋相对，却在同一语域里相安无事。

关　联　护栏、对齐、提示词

护栏

`guardrails` · *hù lán*

字　面　路边、桥侧防坠落冲出的栏杆。喻体来自高速公路。

释　义　约束模型输入输出的安全机制统称：内容过滤、主题白名单、输出格式校验等，常作为模型之外的独立组件实现。

例　句　「这个 agent 能直接操作生产库，护栏必须做在模型外面，光靠系统提示词拦不住。」

> **语言学注**
>
> 仿译 guardrails，说服力全在喻体挑选：护栏不反对车跑，只防坠崖——既承认模型该放开跑，又宣示风险兜得住，比「审查」「封禁」开明，比「安全策略」可感，是一枚为工程与公关双重优化的隐喻。它还完成了一次责任转移：汉语传统劝人「悬崖勒马」，责任在骑手；「护栏」把责任交给修路方，从个体德性挪到基础设施，这是舶来喻体带进汉语的真正新意。工程语域里它已实心化为架构图上的一个框：「护栏层」「护栏规则」，修辞落成了组件。

关　联　越狱、对齐、智能体

检索增强／RAG

`retrieval-augmented generation` · *jiǎn suǒ zēng qiáng*

字　面　「检索」＝查找索取，图书馆学旧词；「增强」＝使更强。全称「检索增强生成」，逐语素对译 retrieval-augmented generation。

释　义　先从外部知识库检索相关材料塞进上下文，再让模型据此作答的架构；用于补知识、抑幻觉、给答案带出处。

例　句　「产品手册三个月一改版，与其反复微调，不如上 RAG，答案还能带引用。」

> **语言学注**
>
> 字母词碾压意译的现在进行时：全称七音节四平八稳，日常交流几乎全被「RAG」取代，还嵌进汉语动词框架——「RAG 一下」「先把文档 RAG 进去」，缩略词直接充当动词词根，是语码转换深入句法层的铁证。「增强」延续 augmented 家族译法（增强现实 AR），构成跨领域的译名呼应。初学者的坑在概念不在词：RAG 不是让模型「学会」新知识，而是开卷考试——「记进脑子」与「摆在桌上」之别，术语字面毫无提示，全靠架构图和讲解补课。

关　联　幻觉、上下文窗口、语料、智能体

预训练

`pre-training` · *yù xùn liàn*

字　面　预先训练。「预」对译 pre-；「训练」本指操练人或牲畜。

释　义　在海量通用语料上以自监督目标（如预测下一个词）先行训练，得到可再微调的通用底座；GPT 的 P 即 pre-trained。

例　句　「预训练语料里代码占比不高，难怪它写 SQL 顺手，写 Verilog 就抓瞎。」

> **语言学注**
>
> 前缀仿译（预-）叠在头号拟人隐喻「训练」之上——把梯度更新说成操练，模型自动领到学员身份，「学习率」「课程学习」「灾难性遗忘」全是这套学堂隐喻的枝叶，汉语用户日用而不觉。「预」字则埋着时间指向的陷阱：所谓「预」是相对微调而言，可预训练恰恰成本最高、耗时最长、定生死，却顶着「预备、热身」的名字——术语的谦辞与工程的重心正好颠倒，初学者极易被字面带偏，以为正菜在后头。

关　联　微调、语料、大模型／小模型、涌现

推理

`inference / reasoning` · *tuī lǐ*

字　面　推究事理，由已知推出未知。喻体在逻辑学与侦探小说（「推理小说」）。

释　义　一词两义：工程语境指模型的前向计算，即「跑模型」（inference），如推理成本、推理框架；能力语境指多步逻辑推导（reasoning），如推理模型、推理能力。

例　句　「推理服务 QPS 上不去，跟模型推理能力强不强是两码事，周会上别混着说。」

> **语言学注**
>
> 一个译名吞下两个英文词的系统性歧义。inference 统计学旧译「推断」，深度学习工程圈却习用「推理」；等 reasoning model 走红，「推理」又被抓去译 reasoning，于是「推理卡」（跑 inference 的 GPU）与「推理模型」（会 reasoning）在同一段话里各说各话——英文两词分工明晰，中文这边语义窄化撞上语义扩张，是译名规划的历史欠账。更妙的是感情色彩：inference 义的「推理」纯属机械前向计算，却借逻辑学光环平添智能想象——把「跑一遍网络」叫「推理」，本身就是一次不动声色的拟人。

关　联　思维链、端侧／本地部署、温度、预训练

端侧／本地部署

`on-device / local deployment` · *duān cè / běn dì bù shǔ*

字　面　「端」＝终端设备一「侧」（与云侧相对），「侧」作方位后缀；「部署」原为军事用语：部署兵力。

释　义　让模型在手机、PC 或企业内网服务器上运行而非调云端 API；卖点是隐私、离线与成本可控。

例　句　「合规要求数据不出内网，云端 API 再便宜也白搭，只能本地部署开源模型。」

> **语言学注**
>
> 「端侧」是汉语自产行话，英文并无单词对应（要说 on-device 或 edge）：「侧」缀构词力旺盛——云侧、端侧、服务侧、业务侧——源自通信业把系统画成两造对望的「侧翼」视角，一个缀就完成架构站位表达，是汉语技术语体自己长出的语法手段。「本地部署」则庄重如基建项目：军事旧词「部署」经 deploy 对译彻底中性化，四字格的郑重恰好托住它的卖点——数据主权。与轻快的「跑在本地」「装一个」相比语域高下立见：销售材料用前者，工程师闲聊用后者。

关　联　量化、蒸馏、大模型／小模型、推理

智能体

`agent / agentic` · *zhì néng tǐ*

字　面　有智能的实体。「体」是科学译词惯用的实体后缀：导体、载体、抗体。

释　义　能自主规划、调用工具、多步执行任务的 LLM 应用形态；agentic 形容「放手让模型自己干」的工作方式。

例　句　「别一上来就搭多智能体，先把单 agent 的工具调用调稳，编排是后话。」

> **语言学注**
>
> agent 是翻译学的老大难：哲学译「能动者」，语言学译「施事」，经济学译「代理人」，化学干脆是「剂」；九十年代多智能体系统研究定下「智能体」，LLM 时代坐收现成。「体」缀选得聪明——避开「代理人」的中间商气味，也比「机器人」少一副躯壳——代价是把 agent 的词根本义 agency（能动性）留在译文之外，「智能」抢了「自主」的戏。口语里字母词与汉译分工渐显：写汇报用「智能体」，改代码说「这个 agent」。至于 agentic 至今无公认汉译（「智能体化」拗口），只能整词嵌入「搞得 agentic 一点」——语码转换填补构词空缺的活例。

关　联　护栏、检索增强、提示工程

大模型／小模型

`large language model (LLM) / small model` · *dà mó xíng ／ xiǎo mó xíng*

字　面　按参数规模给模型分大小，「大／小」是最朴素的度量形容词。

释　义　「大模型」在中文已从「参数量大的模型」漂移为 LLM 乃至生成式 AI 的统称；「小模型」既指传统轻量模型，也指可端侧运行的小参数 LLM。

例　句　「老板嘴里的『上大模型』其实就是接个 API；真正要选型的是哪家、多大参数、要不要再蒸个小的兜底。」

语言学注

中文圈自产的统称，语义比 LLM 更宽：英文 large language model 锚死「语言」，中文省略掉 language，「大」直接修饰「模型」，文生图、多模态遂被口语一并划入——省略造成的语义扩张，如今行业报告都将错就错。「大」在此近乎类前缀（大模型、大数据、大厂），承载「规模即范式」的时代想象；「小模型」靠反义对举获得身份，命名顺序与技术史倒挂——小模型明明先存在，却要以「非大」自居。「百模大战」「卷大模型」入选流行语，说明它早已越出行话进入社会流通层，是本册最出圈的一条。

关　联　涌现、蒸馏、端侧／本地部署、预训练

分词

`tokenization / word segmentation` · *fēn cí*

字　面　切分出词，动宾紧缩：「分」＋「词」。

释　义　把原始文本切成模型可处理的单元序列。传统中文 NLP 指切出语言学意义上的词；LLM 语境指按词表（BPE 等）把文本编码为 token 序列，单位未必是词。

例　句　「一个汉字经常被拆成两三个 token，同样一段话中文就是比英文费 token，分词器的锅。」

语言学注

旧词被新技术偷梁换柱的标本。「分词」原是中文信息处理的招牌难题——汉语不空格，「结婚的和尚未结婚的」怎么切养活过一代研究者；如今 LLM 的「分词」是子词级统计切分，token 常横跨词界甚至劈开单个汉字（按字节切），与语言学的「词」几乎无关，术语却原样沿用，「词」名存实亡。同一个词，语言学家听到词法边界，炼模型的人听到 BPE 词表——这枚化石记录了 NLP 从语言学驱动到统计驱动的范式换血。再添一乱：它与英语语法课的「分词」（participle，现在分词、过去分词）同形撞名，跨语域检索一团浆糊。

关　联　词元、语料、上下文窗口

泛化

`generalization` · *fàn huà*

字　面　「泛」＝广泛、漫溢（泛滥的泛），「-化」为使动后缀：使普遍适用。

释　义　模型把训练中学到的规律用于未见数据的能力；「泛化差」约等于只会背题。

例　句　「换了一家供应商的图纸格式指标就掉一截，模型泛化还是不行。」

语言学注

「-化」缀仿译 -ization 是现代汉语消化西学的主力构词法（工业化、正则化、归一化），「泛化」是其中语义最抽象的一枚：不改变实体，只扩大适用范围。逻辑学早把 generalization 译作「概括」，机器学习却另起炉灶选「泛化」——避开了「概括」的日常义，也牺牲了透明度：初学者难从「泛」联想到「举一反三」，倒容易联想到贬义的「泛泛」。它与「过拟合／欠拟合」构成概念三角却不共享构词模板，这个系统性缺口暗示三词并非同批引进。近年已有反哺用法：「这套打法泛化能力很强」，技术评语被挪去夸方法论了。

关　联　过拟合、欠拟合、预训练

注意力

`attention` · *zhù yì lì*

字　面　心理学的「注意」加能力后缀「力」：把心神集中于一处的能力。

释　义　Transformer 的核心机制，让序列中每个位置按相关度加权地「看」其他位置；部件名有自注意力、注意力头。

例　句　「文档一长注意力就摊薄，关键条款埋在中段容易被漏，就是常说的 lost in the middle。」

语言学注

认知隐喻的巅峰输出：attention 本是心理学描述认知资源分配的术语，进了神经网络实指一套矩阵加权运算，中文沿用心理学旧译「注意力」，拟人深度比「幻觉」更甚——幻觉说的是产出的病症，「注意力」直接宣称模型拥有一种心智官能。「力」缀居功至伟：汉语「X 力」（记忆力、理解力、战斗力）天然把 X 名词化为可增减的资源，「注意力不够用」「摊薄」「聚焦」等搭配全部自动继承。名论文 Attention Is All You Need 的中译在社区流传如格言，更给术语镀了层原典色彩。危险也在此：拿心理直觉去揣摩矩阵乘法，是初学者误解模型机理的第一来源。

关　联　上下文窗口、思维链、涌现

思维链

`chain-of-thought (CoT)` · *sī wéi liàn*

字　面　把思维串成链条。喻体是锁链：一环扣一环，断一环则全断。

释　义　让模型先输出逐步推导再给答案的提示技巧及相应输出形态；推理模型内置的长思考过程也叫思维链。

例　句　「加一句『一步一步想』，思维链展开之后这类应用题正确率直接翻倍。」

> **语言学注**
>
> 严格仿译 chain-of-thought，且缩得比原文漂亮：英文靠介词搭骨架，中文压成三字格，又赶上「产业链」「区块链」以降「X 链」词族的东风，落地即有术语相。「链」的隐喻自带线性与因果承诺——每环由前环导出——这既是卖点也是靶心：批评者指出模型完全可能先有答案再补一条好看的链，「思维」与真实计算未必对应。词族随隐喻繁殖：思维树、思维图，框架一立，造词按图索骥。拟人层面它与「注意力」同宗：把生成的中间文本径称「思维」，又一次悄悄预设了心智。

关　联　推理、提示工程、注意力

炼丹

alchemy（本土隐喻，非译词）　·　*liàn dān*

字　面　道教方士烧炉炼制长生丹药。喻体是丹炉：投料、控火、封炉、开炉，成败半凭天意。

释　义　圈内俚语，戏称训练与调参的日常：GPU 集群是「丹炉」，数据是药材，超参配置是「丹方」，训崩了叫「炸炉」，出好模型是「出仙丹」；「炼丹师」为算法工程师自称。

例　句　「卡终于到位，今晚开炉，明早看 loss——但愿这回别炸炉。」

> **语言学注**
>
> 本册唯一不靠翻译、纯从本土文化库里请出来的隐喻，与英文社区的自嘲 alchemy（学界曾激辩「机器学习是炼金术」）不谋而合又各烧各的炉：炼金术求点石成金，炼丹求长生不老，共同点是有配方、无理论、难复现——自嘲正中深度学习的方法论痛处。修仙语汇与加班现实的语域反差完成情绪泄压，且能产性惊人：丹炉、丹方、开炉、炸炉、金丹，一个喻体撑起整套黑话。它还是身份暗号：能把「炼丹」说得眉飞色舞的，多半真守过炉。

关　联　调参、微调、收敛

调参／调参侠

hyperparameter tuning　·　*tiáo cān ／ tiáo cān xiá*

字　面　「调参」＝调节参数，动宾紧缩；「侠」＝武侠之「侠」，此处纯属反讽。

释　义　调整学习率、批大小等超参数以改善训练效果的工作；「调参侠」自嘲只会盲调参数、不明原理也无力创新的算法工程师。

例　句　「面试背了一肚子八股，上手连学习率都不会定，连合格的调参侠都算不上。」

> **语言学注**
>
> 「调参」是标准动宾紧缩行话（同型：压测、联调、扩容），感情中性；缀上「侠」立刻翻转为苦涩自嘲——武侠语域的尊称嫁接到重复劳动上，语域反差生成反讽，同族有「CRUD 侠」「if-else 侠」，「X 侠＝只会干 X 的人」是网络汉语后缀化的鲜活案例，「侠」的本义侠义被彻底掏空。这对词还记录了一段行业心态史：红利期人人自称炼丹师，内卷期自贬为调参侠——同一份工作的两个绰号，一个往仙侠里抬，一个往流水线上踩。

关　联　炼丹、微调、过拟合

喂数据／喂给模型

`feed (data)` · *wèi shù jù*

字　面　「喂」＝给人或动物送食。喻体来自饲养：数据是饲料，模型是那张永远吃不饱的嘴。

释　义　向模型或训练管线输入数据的口语说法，训练（喂语料）与调用（把文档喂给模型）两个场景通用。

例　句　「把整份 API 文档喂给它再让它写调用代码，比让它凭记忆瞎猜强多了。」

> **语言学注**
>
> 对译英文 feed，但汉语「喂」的饲养义更露骨：英文 feed 尚有「馈送」的机械义（feeder、feed line），「喂」几乎只剩喂饭喂猪，拟人（或者说拟畜）色彩反比原文更浓。一个「喂」字带活整条消化道隐喻：「吃进去」「消化不了」「吐出来」——「这批数据它消化不良，全过拟合了」。语域分工干脆利落：口语说喂，论文写「输入」「注入」。与近义的「灌数据」还有一丝感情差：喂是细心投喂，灌是不由分说——一字之差，写尽数据管线旁的两种姿态。

关　联　语料、预训练、检索增强

附录 · 江湖与工位：黑话拾遗

正文八辑收的是技术概念。但这门方言里还有一支，说的不是系统，而是**说系统的人**——他们的处境、情绪与自嘲。这类词几乎全是本土原创、纯口语、进不了正式文档，却最能泄露行业的集体心态。谨拾遗数条，语域一律标注为「戏谑／黑话」，正式场合勿用。

删库跑路

`drop the database and run` · *shān kù pǎo lù*

字　面　把数据库删了，然后跑路（辞职逃走）。

释　义　程序员闯下大祸或愤懑至极时的终极想象——`DROP DATABASE` 之后一走了之；多为玩笑，偶尔是真实事故的复盘标题。

例　句　「权限收一收吧，别哪天谁一言不合删库跑路，备份都来不及恢复。」

> **语言学注**
>
> 两个动宾短语连动成一句微型犯罪叙事：「删库」（动宾紧缩）＋「跑路」（市井旧词，本指欠债潜逃）。它是行业创伤记忆的语言化石——正因真发生过几起，玩笑才有了牙齿；同时又是一句反向安全布道，每次出口都在替「最小权限」和「备份」敲钟。

关　联　最小权限、回滚、背锅侠

提桶跑路

`grab the mop bucket and quit` · *tí tǒng pǎo lù*

字　面　拎起（保洁的）水桶就走人。画面取自工地或食堂——卷铺盖不够狼狈，索性连桶都拎上。

释　义　辞职或被裁的戏称，尤指一气之下或走投无路的离场；与「删库跑路」共享「跑路」词根，但这一支不搞破坏，只搞退场。

例　句　「需求又双叒改了第五版，兄弟们，桶都给你们备好了。」

> **语言学注**
>
> 「跑路」这一市井语素在开发圈裂变出一个词族（删库跑路、提桶跑路），靠更换前面的动宾来调节语气——一个搞破坏、一个认命。「提桶」的具体道具感是关键：抽象的「离职」被一只水桶拽回地面，卑微与幽默同时到位，是网络汉语「以实写虚」的典型。

关　联　内卷、背锅侠、甩锅

内卷

`involution` · *nèi juǎn*

字　面　向内卷曲——不向外扩张，反而在内部越缠越紧。

释　义　同行间无实际增益的过度竞争：别人加班你也得加，卷进去谁都没多得好处，只是把标准抬高、把人耗干。

例　句　「这需求本来一周的活儿，有人主动周末赶，一卷，全组的排期基线就没了。」

> **语言学注**
>
> 难得的「学术词出圈」标本：involution 本是人类学（格尔茨论爪哇农业）与数学的术语，2020 年前后经社交网络语义漂白、爆火，成为全民口头禅，技术圈只是重灾区之一。它填补了一个真实的表达空缺——「零和的过度竞争」汉语原先没有单词——因而禁而不绝；代价是语义稀释到近乎万能，什么竞争都能叫内卷。

关　联　提桶跑路、冲刺、造轮子

面向搜索引擎编程

`search-engine-oriented programming` · *miàn xiàng sōu suǒ yǐn qíng biān chéng*

字　面　仿「面向对象编程」（object-oriented programming）的正经术语，把「对象」换成「搜索引擎」。

释　义　调侃式的开发方法论——不靠自己想，靠搜：遇事先搜、照抄改改、跑通即可。同族还有「面向复制粘贴编程」「面向工资编程」「面向简历编程」。

例　句　「这段异步锁的写法我也没全懂，面向搜索引擎编程抄来的，能过测试先用着。」

> **语言学注**
>
> 一次精准的**仿格造词**：借「面向 X 编程」这个正经范式的空槽，填进一个反差极大的宾语，庄重的术语外壳与心虚的实情碰撞出反讽。「面向 X 编程」由此升级为能产格式，X 越离谱越好笑（工资、简历、老板、KPI），是网络汉语用「戏仿权威句式」制造幽默的活样本，也含着对「工程师专业性」的温柔自嘲。

关　　联　造轮子、银弹、屎山

词条索引

按辑排列，点击跳转。共 235 则。

第一辑 · 故障与可靠性

第二辑 · 测试与质量

第三辑 · 架构与设计

屎山（代码）
祖传代码
上帝对象／上帝类
贫血模型
充血模型
六边形架构／端口与适配器架构
洋葱架构
单体／巨石应用
微服务
分布式单体
服务网格
中台
门面／外观（Facade）
接缝
防腐层
绞杀者模式／绞杀榕模式
反模式
黄金路径／快乐路径
关键路径
造轮子／重复造轮子
内聚
耦合
正交
抽象泄漏／漏水的抽象
可插拔／插件化

第四辑 · 流程与协作

提交
合入／合并
冲突／合并冲突
变基
拣选／樱桃挑选
压平／拍平／squash 合并
主干
特性分支
拉取请求（PR）／合并请求（MR）
代码评审／代码审查（CR）
回退／还原
重置（reset --hard）
回滚
流水线
持续集成（CI）
持续交付／持续部署（CD）
上线
热修复
打补丁
封版／代码冻结
里程碑
排期
冲刺
积压／待办
看板
站会／每日站会
无责复盘
联调
提测
自测
对齐／拉齐
甩锅
背锅侠

第五辑 · 性能与容量

吞吐／吞吐量
延迟
长尾
热点／热键
冷启动
预热／热身／warm-up
削峰填谷
压测／负载测试／压力测试
基准测试／跑分／benchmark
瓶颈
打点
埋点
水位／水位线／watermark
令牌桶
漏桶
QPS／TPS
尾延迟／P99／P999
毛刺
抖动
卡顿
扩容／缩容

第六辑 · 数据与并发

分片
分库分表
读写分离

主从／主备（主副本-从副本）
回源
落盘
刷盘
脏页
快照隔离
乐观锁
悲观锁
自旋锁
读写锁
心跳
探活（存活探针／就绪探针）
保活
优雅关闭／优雅退出
脏读
幻读
不可重复读
生产者-消费者
幂等键
最终一致性
强一致性
写扩散／读扩散
双写
扇入／扇出
脑裂
墓碑
写时复制

第七辑 · 安全与运维

堡垒机
跳板机
蜜罐
沙箱／沙盒
哨兵
看门狗
探针／存活探针／就绪探针
影子／影子库／影子流量
蓝绿部署
金丝雀发布
灰度发布
最小权限
纵深防御
零信任
加盐
脱敏
越权
提权
巡检
值班／on-call
告警风暴
根因分析／根因／RCA
撞库
后门
红蓝对抗／红队／蓝队
复盘

第八辑 · AI 与大模型时代

语料
词元／令牌／token
微调
蒸馏
量化
幻觉
提示词／提示
提示工程
上下文窗口
对齐
涌现
过拟合
欠拟合
召回率
精确率
收敛
温度
越狱
护栏
检索增强／RAG
预训练
推理
端侧／本地部署
智能体
大模型／小模型
分词
泛化
注意力
思维链
炼丹
调参／调参侠
喂数据／喂给模型

附录 · 江湖与工位：黑话拾遗

编纂说明。本辞典把中文开发者（与对话式 AI）日常使用、却在通用汉语中罕见的行话，当作一门「行业社会方言」加以整理：每条给出字面、释义、真实例句，以及最要紧的「语言学注」——它的构词机制、语域与感情色彩。

词条由八位 Claude **Fable** 子代理按开发领域分辑撰写，经语言学编者统稿、去重、并加序言与附录。收词力求覆盖故障、测试、架构、流程、性能、数据并发、安全运维与 AI 八大场景，非穷尽，亦难免挂漏。释义与考据仅供语言学赏玩，不作规范依据。