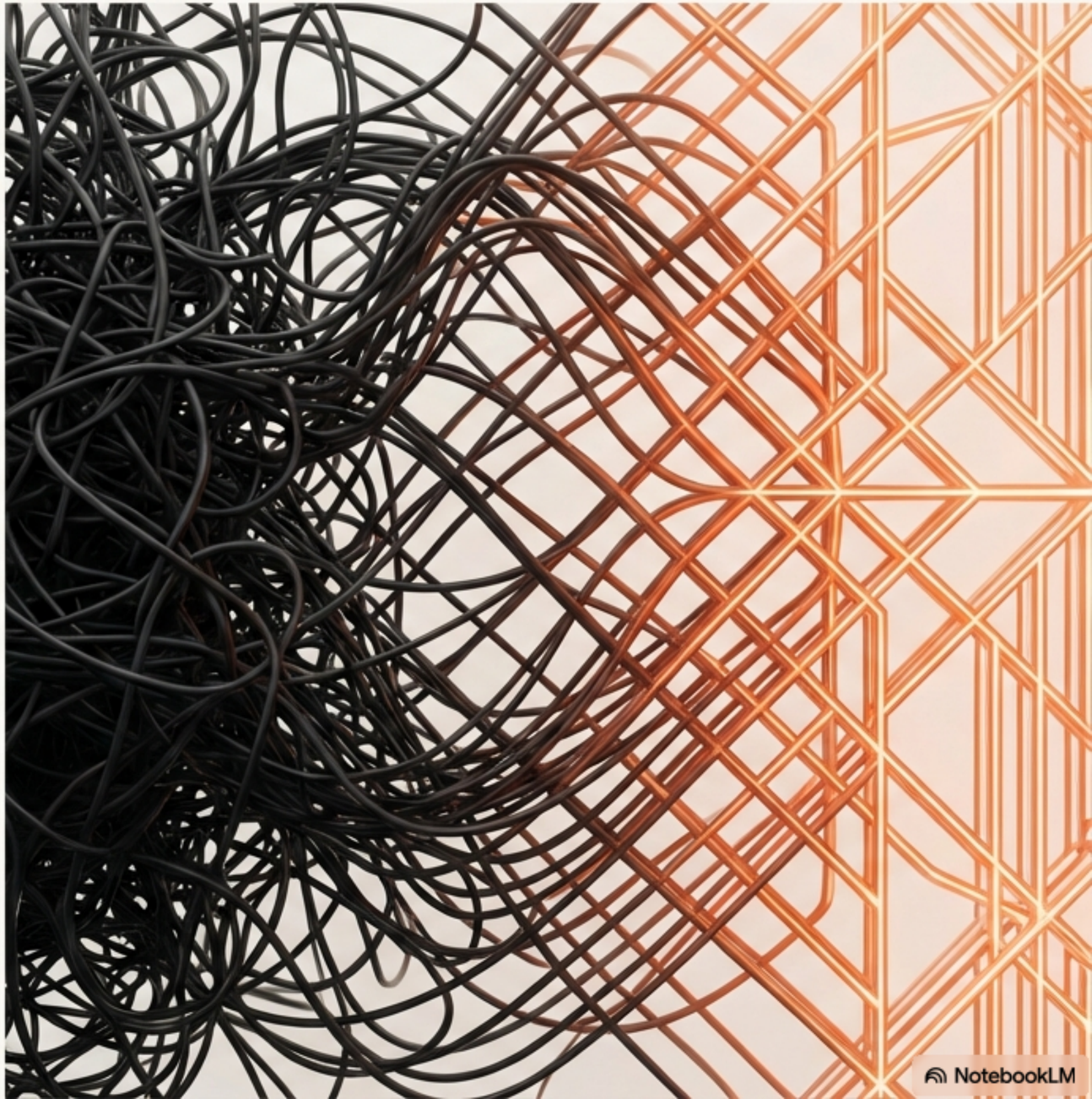
The Art of the Short Link

From Flawed Hacks to Elegant Engineering



The Digital Mile

`ID: 9,223,372,036,854,775,807`

`URL: https://s.io/92233720368547758O7`

Raw database IDs create URLs that are long, ***ugly, and bad for user experience.***

Neue Haas Grotesk Display Pro: The Power of a Bigger Alphabet

The more symbols a number system has, the fewer characters you need to represent the same value.

The Obvious First Step: Encode It

Introduce Base64 as the go-to solution for encoding.

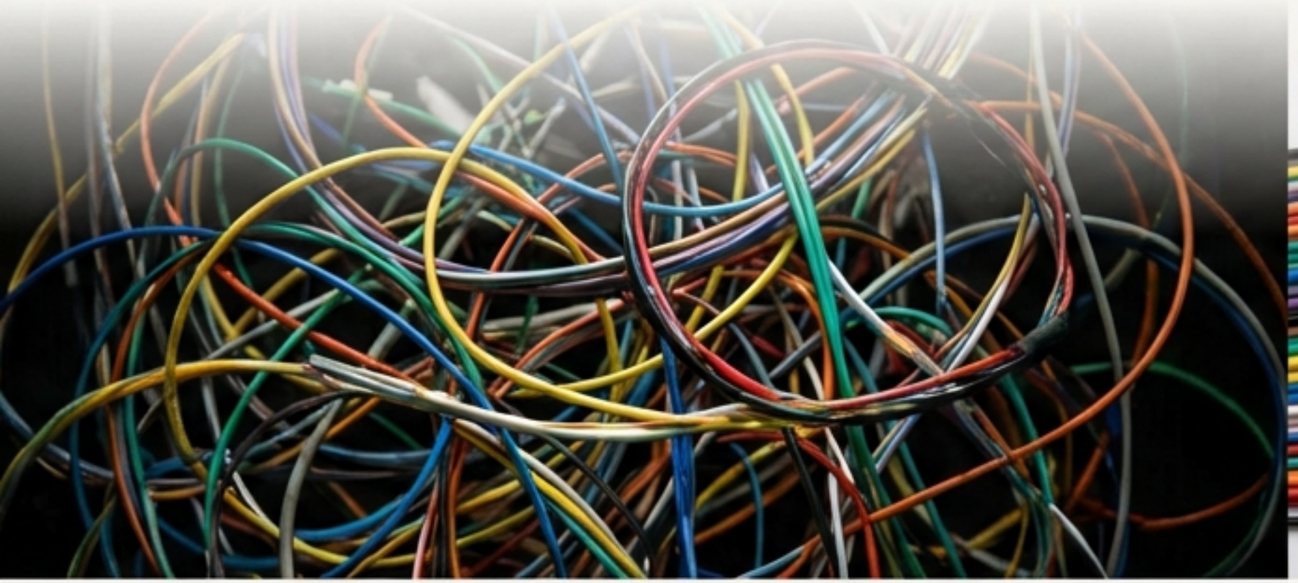
`A–Z a–z 0–9 + /`

Encoding	Approx length
Decimal (base-10)	19 chars
Base64	~11 chars

From Unwieldy to Elegant

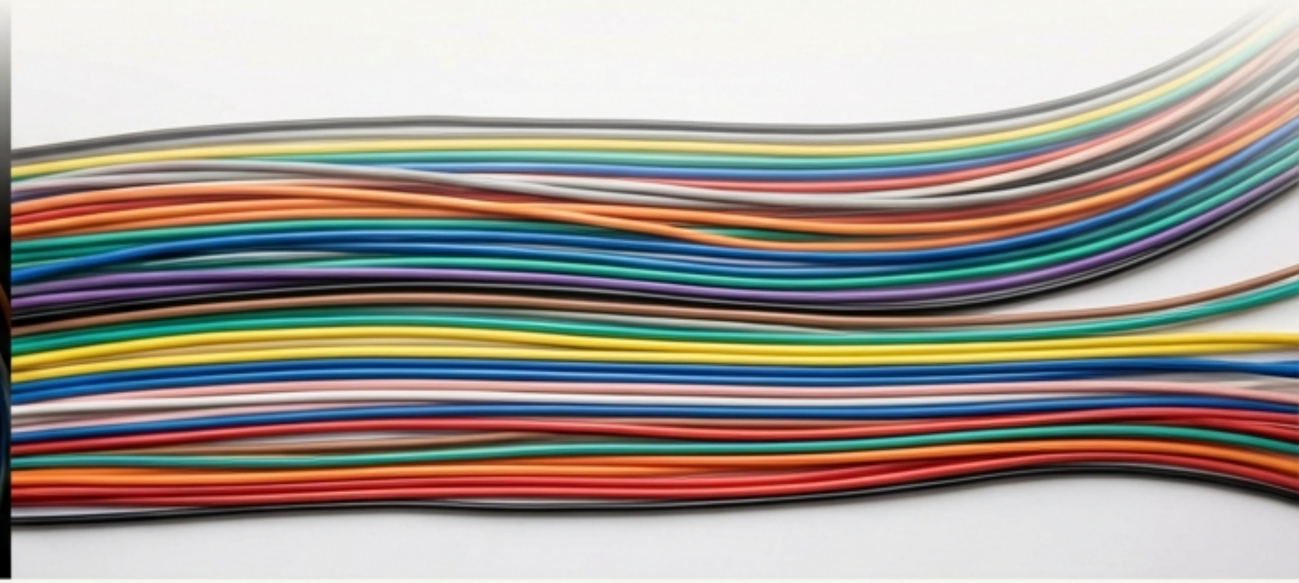
Before

https://s.io/92233720368547758O7



After

https://s.io/**gP9QK2D2z0f**



The URL is shorter. The UX is better. Problem solved... right?

The Hidden Danger in Plain Sight

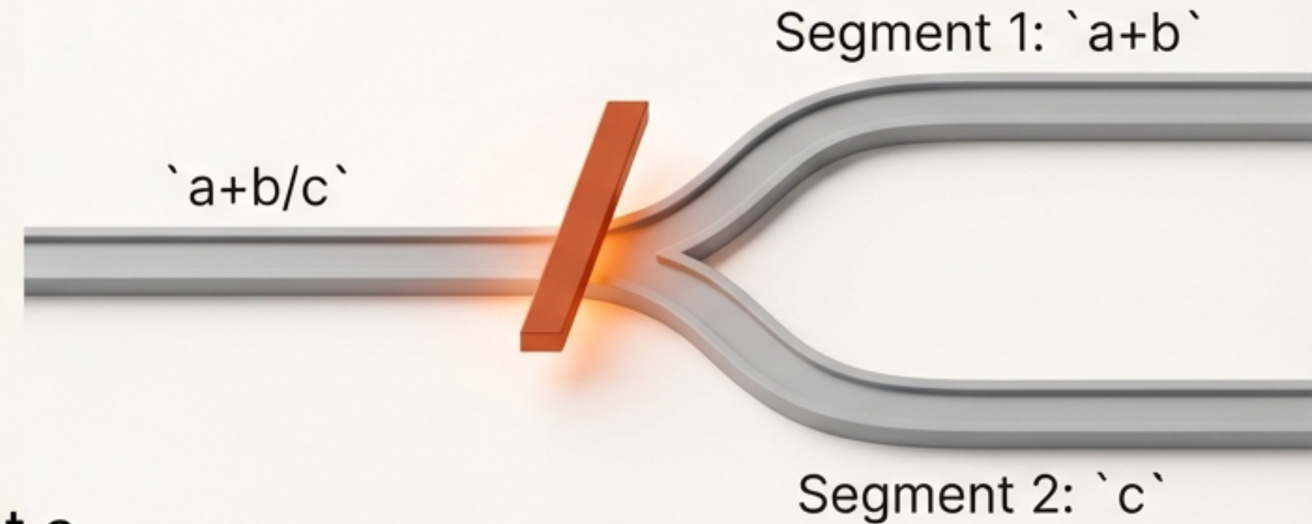
These two symbols are not just characters; they have special, reserved meaning within the structure of a URL.

The Slash: A Path Separator

A URL is not a flat string. A URL like `https://s.io/a+b/c` is parsed by servers into distinct segments:

* Segment 1: `a+b`
* Segment 2: `c`

This breaks routing systems that expect a single identifier, like `GET /{shortCode}`, because the router now sees two segments.



The Plus: A Character of Many Faces

Explain the contextual ambiguity of the plus sign.

- In URL query strings, `+` is commonly interpreted as a ‘space’.
- A path like `/a+b` can be decoded by different servers, proxies, or frameworks as "a b".
- This leads to inconsistent behavior and hard-to-debug errors.

The Ugly Compromise: URL Encoding

`a+b/c` becomes `a%2Bb%2Fc`
The resulting URL is: `https://s.io/a%2Bb%2Fc`

You've traded one problem for another. The URL is now technically safe but is **long, ugly**, and defeats the entire purpose of creating a short link.

What We Actually Need
Neue Haas Grotesk Display Pro

- ✓ Short Length
- ✓ URL-Safe Characters
- ✓ No Encoding Required
- ✓ A Single, Unbroken Path Segment

The Right Tool for the Job: Base62

`A–Z a–z 0–9`

Highlight what's missing:
No `+`, no /`. No special characters.

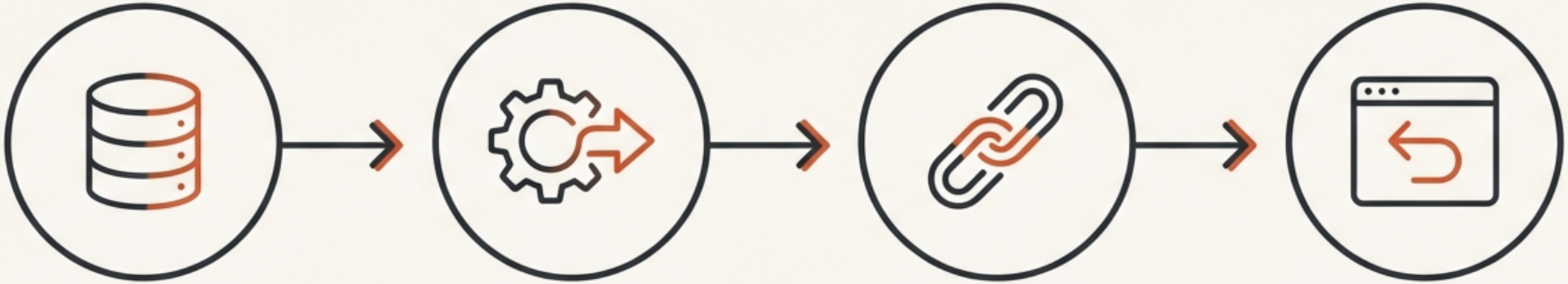
Base62 was invented specifically for **human-facing identifiers** inside URLs.

The Promise, Truly Fulfilled

ID: 9,223,372,036,854,775,807
URL: https://s.io/**AzL8n0Y58m7**

Clean. Short. Safe. No compromises.

The Elegant Flow



GENERATE ID

A database generates a **BIGINT** primary key.

ENCODE

An encoding engine converts the ID to a **Base62 string**.

EXPOSE URL

The Base62 string is exposed as the **short URL path**.

DECODE & FETCH

On request, the server decodes the string back to the **BIGINT ID** to find the original record.

How Base62 Solves Our Core Problems

URL-Safe Characters?

Uses only `[0-9][A-Z][a-z]`. No special characters like `/`, `+`, `%`, or `=`.

Short Length?

~11 chars for a max BIGINT, the sweet spot for usability.

No Encoding Needed?

Always results in a single, clean path segment that just works.

An Honest Assessment

❌ **Predictability:** The encoded IDs are still sequential (`125` → `cb`, `126` → `cc`, `127` → `cd`), which allows for enumeration.

❌ **Database Coupling:** The solution still exposes a form of your internal primary key, tying the URL format to your database strategy.

⚠️ **Case Sensitivity:** This is a trade-off. It increases the number of possible characters but can ɔn lead to human error (`⟨u\' `a is not the same as `A).

When to Choose Base62

Base62 is a strong, practical default when you need short, collision-free, URL-safe identifiers and can manage the risk of predictability through other means like access control or rate limiting.

The Story So Far

The goal is **short, human-readable**, and **technically clean** URLs.

Standard **Base64** is a **trap** due to unsafe characters (`**+**', `**/**') that break URL routing.

Using **URL-encoding** to 'fix' Base64 makes links ugly and defeats the original purpose.

Base62 is the **purpose-built, URL-safe** encoding scheme designed for this specific engineering problem.

Engineering is the Art of the Right Compromise

Choosing the right encoding is not about finding a magic bullet, but about understanding the constraints of the system—from server routing to human psychology—and selecting the tool that fits the job. Base62 is not perfect, but it is a deliberate ɾan and elegant solution to a well-defined problem.