

기술

검사 결과가 0건일 때, 없는 것과 검사가 고장 난 것을 어떻게 가릅니까

검사를 돌렸는데 결과가 0건으로 나왔습니다. 이때 우리는 보통 안심합니다. 걸린 게 없다는 뜻이니까요. 그런데 0은 두 가지를 동시에 뜻합니다. 정말 없거나, 검사가 고장 났거나. 이 둘은 화면에서 똑같이 생겼습니다. 오류 메시지도, 빨간 글자도, 종료 코드 1도 없습니다. 그냥 조용히 0입니다.

방동걸 (주)웨이스 대표이사

2026년 9월 7일

검사를 돌렸는데 결과가 0건으로 나왔습니다. 이때 우리는 보통 안심합니다. 걸린 게 없다는 뜻이니까요. 공장 운영체제 VEXPLOR를 만드는 웨이스도 사내 검사 도구에서 그 안심이 틀렸던 적이 있어 적었습니다.

그런데 0은 두 가지를 동시에 뜻합니다. 정말 없거나, 검사가 고장 났거나. 이 둘은 화면에서 똑같이 생겼습니다. 오류 메시지도, 빨간 글자도, 종료 코드 1도 없습니다. 그냥 조용히 0입니다.

저희는 지난주에 이 0을 그대로 믿을 뻔했습니다. 그대로 믿었으면 정반대 결론을 문서에 적고 끝냈을 겁니다.

01

사건: 20건이 전부 "목록에 없음"으로 나왔습니다

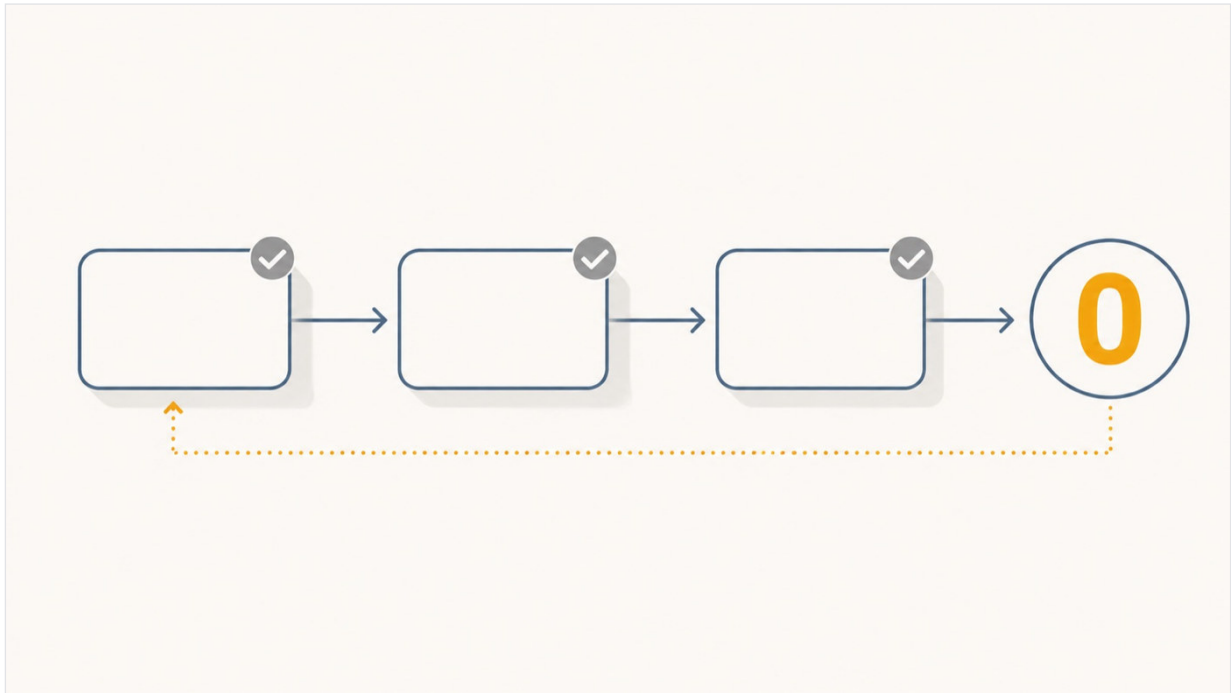
검색엔진 색인 리포트에서 URL 20개가 "발견했지만 아직 수집하지 않음" 상태로 잡혔습니다. 원인을 좁히려면 먼저 물어야 할 게 있습니다. 이 URL들이 우리 사이트맵에 실려 있기는 한가?

사이트맵을 내려받아 URL 목록을 뽑고, 20건이 그 목록에 있는지 한 줄씩 대조했습니다. 결과는 이랬습니다.

```
/about code=200 noindex=0 사이트맵=0
/privacy code=200 noindex=0 사이트맵=0
/tag/검색 code=200 noindex=0 사이트맵=0
...
```

20건 전부 사이트맵=0. 페이지는 살아 있는데(200) 사이트맵에는 하나도 없다. 깔끔한 진단으로 보였습니다. "사이트맵에 없으니 수집이 늦는 것이다, 사이트맵 생성 로직을 봐야 한다."

틀렸습니다. 20건 전부 사이트맵에 있었습니다.



세 단계가 모두 통과했는데 결과는 0이었습니다. 실패한 단계가 없으니 실패를 알 방법도 없었습니다.

02

0을 만든 것은 데이터가 아니라 정규식 한 글자였습니다

사이트맵에서 URL을 뽑는 코드는 이랬습니다.

```
grep -o '<loc>[^<]*</loc>' | sed 's|</?loc>||g'
```

`</?loc>` 는 "`</loc>` 또는 `<loc>`"을 뜻하려고 쓴 것입니다. `?` 는 앞 문자가 있어도 되고 없어도 된다는 표시니까요.

GNU sed에서는 그렇게 동작합니다. macOS의 BSD sed에서는 아닙니다. BSD sed의 기본 정규식에는 `?` 가 없어서, 저 패턴은 물음표 문자 자체를 찾습니다. 사이트맵에 물음표가 붙은 `</?loc>` 같은 문자열은 없으니 아무것도 치환되지 않습니다.

그래서 파일에는 이런 줄이 저장됐습니다.

```
<loc>https://example.com/about</loc>
```

태그가 그대로 남은 줄입니다. 여기에 `grep -xF "https://example.com/about"` 로 정확히 일치하는 줄을

찾으면 당연히 0건입니다. 데이터는 멀쩡한데 형식이 어긋난 것이고, 검사는 그 차이를 "없음"으로 보고했습니다.

여기서 중요한 건 sed 이야기가 아닙니다. **에러가 하나도 나지 않았다는 점**입니다.

단계	실제로 일어난 일	화면에 나온 것
치환	아무것도 안 바뀜	종료 코드 0
저장	태그 붙은 줄 저장	파일 생성됨
대조	형식 불일치	사이트맵=0

세 단계 모두 "성공"입니다. 실패한 단계가 없으니 실패를 알 방법도 없었습니다.

03

그것을 잡은 것은 재검사가 아니라 다른 방식의 집계였습니다

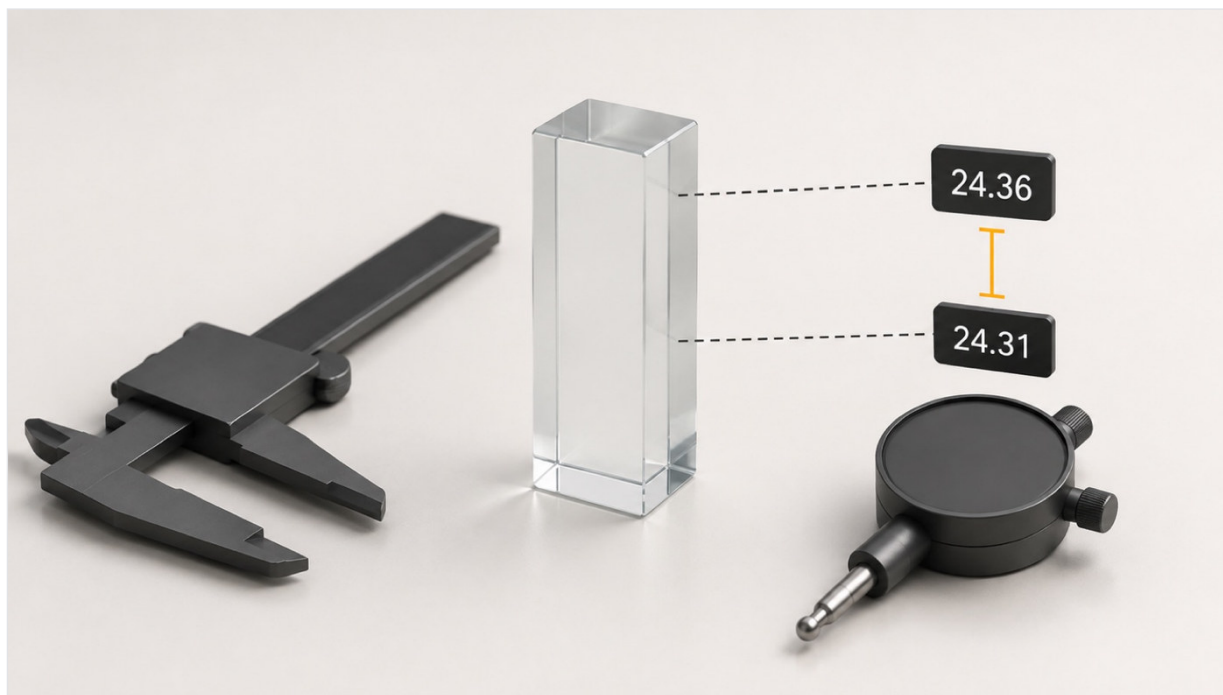
같은 파일로 다른 것도 세고 있었습니다. 사이트맵이 어떤 URL로 이루어졌는지 보려고 경로 첫 조각을 잘라 세는 집계였습니다. 그 출력이 이렇게 나왔습니다.

```
226 <loc>tag
106 <loc>post
1 terms<
1 privacy<
1 about<
```

`<loc>tag` 와 `about<`. 꼬리에 태그 조각이 붙어 있습니다. 그리고 **about** 이 목록에 있습니다. 방금 "사이트맵에 없다"고 판정한 바로 그 `/about` 입니다.

두 결과가 서로 모순됩니다. 하나는 없다고 하고 하나는 있다고 합니다. 둘 다 같은 파일에서 나왔으니 데이터가 문제일 리 없고, 남는 건 읽는 쪽입니다.

이게 실제로 작동한 안전장치였습니다. 재검사가 아니었습니다. 같은 검사를 다시 돌렸으면 같은 0이 다시 나왔을 겁니다. **한 데이터를 서로 다른 방식으로 두 번 센 것이 어긋남을 드러냈습니다.**



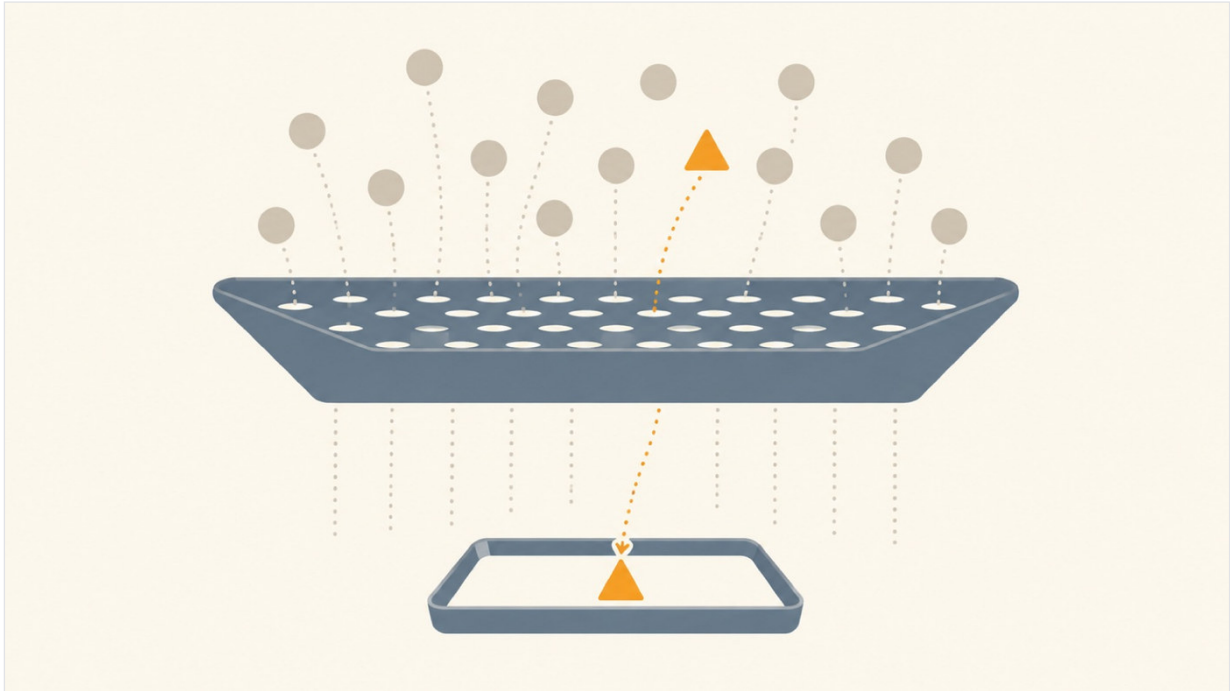
같은 대상을 두 방식으로 재면 값이 어긋납니다. 그 어긋남이 이번에 작동한 유일한 경보였습니다.

04

0을 믿어도 되는지 가르네 가지

이 일에서 규칙으로 남긴 것들입니다.

- **전건 0은 특별 취급합니다.** 몇 건 걸리다가 0이 된 것과 처음부터 전건 0인 것은 다릅니다. 후자는 "대상이 없음"과 "검사가 대상을 못 찾음"이 겹쳐 보이는 값입니다. 결론을 내리기 전에 검사기를 먼저 의심합니다.
- **양성 대조군을 하나 넣습니다.** 반드시 걸려야 하는 입력을 하나 섞어 넣고, 그것이 걸리는지 봅니다. 그게 안 걸리면 0은 데이터가 아니라 검사기의 상태입니다.
- **추출 직후 첫 세 줄을 눈으로 봅니다.** head -3 한 번이면 <loc> 가 남아 있는 걸 즉시 볼 수 있었습니다. 중간 산출물을 보지 않고 최종 판정만 읽은 것이 이번 사고의 실질 원인입니다.
- **같은 데이터를 다른 방식으로도 셉니다.** 개수, 분포, 표본. 하나가 틀리면 나머지와 어긋나서 드러납니다. 이 어긋남이 유일하게 작동한 경보였습니다.



반드시 걸려야 하는 입력을 하나 섞어 넣고, 그것이 걸리는지 봅니다.

05

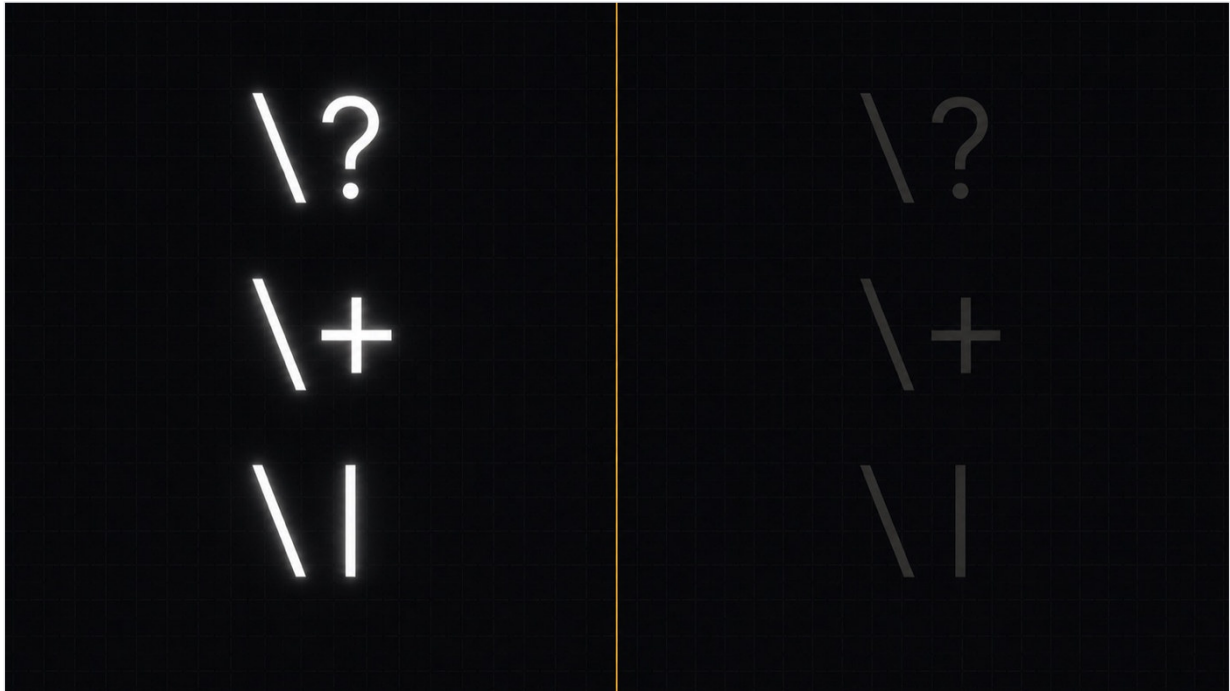
플랫폼 차이는 조용한 오답의 흔한 원인입니다

정규식은 특히 그렇습니다. 문법이 어긋나면 대개 오류가 아니라 **다른 뜻으로 해석**되기 때문입니다.

표기	GNU sed	BSD sed(macOS)
<code>\?</code>	0회 또는 1회	물음표 문자
<code>\+</code>	1회 이상	더하기 문자
<code> </code>	또는	세로줄 문자

세 경우 모두 오류 없이 통과하고, 다만 아무것도 안 걸립니다. 팀에 맥과 윈도우와 리눅스가 섞여 있으면 **같은 스크립트가 사람마다 다른 답을 내는데 아무도 모릅니다.**

피하는 방법은 단순합니다. `sed -E` 로 확장 정규식을 쓰거나, 치환이 목적이면 `grep -o` 나 파이썬으로 뽐습니다. 셸 정규식으로 구조화된 문서(XML·JSON)를 파싱하지 않는 것이 더 근본적인 답이고요.



같은 표기가 한쪽에서는 동작하고 다른 쪽에서는 아무것도 하지 않습니다. 오류는 나지 않습니다.

06

같은 날 나온 다른 0은 믿었습니다

여기서 오해가 생기기 쉽습니다. 0을 전부 의심하면 아무것도 못 넘어갑니다.

같은 작업에서 0이 하나 더 나왔습니다. "사이트맵에 문제 URL 25건이 남아 있는가"를 확인한 결과가 0건이었습니다. 이 0은 그대로 믿고 넘어갔습니다.

차이는 그 검사가 무언가를 잡아 본 적이 있다는 것이었습니다. 같은 검사기가 몇 주 전에 실제로 죽은 URL을 찾아내 고치게 만든 이력이 있었고, 합성 입력 8종으로 고장을 재현하는 자가검증도 붙어 있었습니다.

자가검증이라고 하면 거창해 보이지만 실제로는 이런 목록입니다.

넣는 입력	기대	확인하는 것
죽은 URL	잡힌다	탐지가 산다
정상 목록	조용하다	오탐이 없다
빈 입력	0을 낸다	빈 값 처리
형식 오류	오류를 낸다	조용히 안 넘어간다

네 줄짜리 표지만, 이게 있으면 그 검사의 0은 신호가 되고 없으면 그냥 침묵입니다. 앞의 사이트맵 파싱에는 이게 없었습니다. 한 번도 무언가를 잡아 본 적 없는 검사가 처음 낸 0이었습니다.

그래서 판정 기준을 이렇게 둡니다. 이 검사가 지금까지 몇 번 걸렸는지, 마지막으로 걸린 게 언제인지. 한 번도 안 걸린 검사의 0은 아직 아무것도 아닙니다.

07

검사기를 고칠 것인가, 문장을 바꿀 것인가

같은 주에 비슷한 일이 하나 더 있었습니다. 사내 금지 표현 검사기가 "망가진다"라는 낱말을 잡았습니다. 규칙이 찾는 것은 영어 have를 직역한 표현이었는데, 왼쪽 경계를 두지 않아서 "망가진다"의 뒷부분과 글자가 겹친 겁니다.

선택지가 둘이었습니다. 그 문장에서 "망가진다"를 다른 말로 바꾸거나, 검사기의 정규식을 고치거나.

문장을 바꾸는 쪽이 30초면 끝납니다. 그런데 그러면 다음 사람이 "망가진다"를 쓸 때 같은 경고를 다시 받습니다. 그리고 그 사람도 30초로 피해 갈 겁니다.

가짜 경고가 쌓이면 결국 아무도 검사를 안 봅니다. 검사기가 매일 아침 틀린 소리를 하면, 진짜 경고가 그 사이에 섞여도 안 읽힙니다. 그래서 정규식에 왼쪽 경계를 넣고, 검증 세트에 "망가진다"가 들어간 자연스러운 문장 두 개를 음성 사례로 추가했습니다. 재현율은 50/50 그대로, 오탐은 0/70입니다.

검사기를 고치는 데 든 시간은 10분이었습니다. 문장을 바꿨다면 30초였겠지만, 그 30초를 앞으로 계속 냈을 겁니다.

08

정리

0은 값이 아니라 질문입니다. "없다"와 "못 찾았다"가 같은 자리에 앉아 있고, 둘을 가르는 건 그 0을 만든 경로를 되짚는 일뿐입니다.

검사를 새로 만들 때 저희가 지키려는 순서는 이렇습니다. 먼저 고장 난 입력을 넣어 검사가 그것을 잡는지 봅니다. 그다음 정상 입력을 넣어 조용한지 봅니다. 둘 다 확인한 뒤에야 그 검사의 0을 신호로 삼습니다. 이 순서를 건너뛰면, 통과했다는 것만으로는 아무것도 알 수 없습니다.

앞서 저희가 테스트가 전부 초록인데 사용자가 먼저 문제를 발견한 이야기와 검사 스크립트가 자기 성적을 잘못 보고한 이야기를 적은 적이 있습니다. 이번 건은 그 계열의 세 번째입니다. 세 번 다 공통점이 하나였습니다. 검사가 통과한 것과 대상이 정상인 것은 다른 문장입니다.

이 사고가 실제로 어떤 작업 중에 났는지는 [저희 블로그 태그 226개 중 163개가 글 한 편짜리였다는 이야기에](#) 적었습니다. 위 0은 그 작업의 첫 단계에서 나온 값입니다.

여러분의 파이프라인에서 지금 0을 내고 있는 검사는 몇 개입니까? 그중 몇 개가 마지막으로 무언가를 잡아 본 적이 있습니까?

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.



원문 blog.vexplor.com/post/how-to-tell-zero-from-broken
발행 2026년 9월 7일 · 방동걸 (주)웨이스 대표이사

회사 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 ·
사업자등록번호 439-87-02852
주소 (18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동)
1702·1706호 (영천동, 실리콘엘리동탄)
연락처 전화 070-4730-0629 · contact@wace.me
홈페이지 wace.me · linkedin.com/company/wace-inc



원문



홈페이지



링크드인