

Vim의 바깥에서

Vim을 자유롭게, 소모품처럼 사용하기

created: 2024.12.15

updated: 2024.12.15

[blame 보기](#) / [편집하기](#) / [의견 남기기](#)

상위 문서: [wiki](#) / [Articles](#)

#vim-enter-발표자료

- [일러두기](#)
- [2019년 'Vim, 두 가지 관점' 발표 이후 생각한 것들](#)
 - [LSP, COC 이후로 Vim 플러그인의 제작 패러다임이 바뀌었다](#)
 - [늘어만 가는 복잡도](#)
 - [내가 Vim에 바라는 것들](#)
- [Vim은 그냥 하나의 UNIX 도구](#)
 - [vised](#)
- [주석](#)
- [참고문헌](#)

• 일러두기

이 글은 2024-12-21 [VimEnter 2024 모임](#) 발표를 위해 작성한 것입니다.

• 2019년 'Vim, 두 가지 관점' 발표 이후 생각한 것들

2019년, VIMRC 2019 모임에서 [Vim, 두 가지 관점](#)이란 주제로 발표를 했다.

발표의 내용은 크게 두 가지 관점에서 Vim을 바라보는 것이었다.

1. 관점: 텍스트 처리 언어로서의 Vim
2. 관점: 플랫폼으로서의 Vim

∴ *LSP, COC 이후로 Vim 플러그인의 제작 패러다임이 바뀌었다*

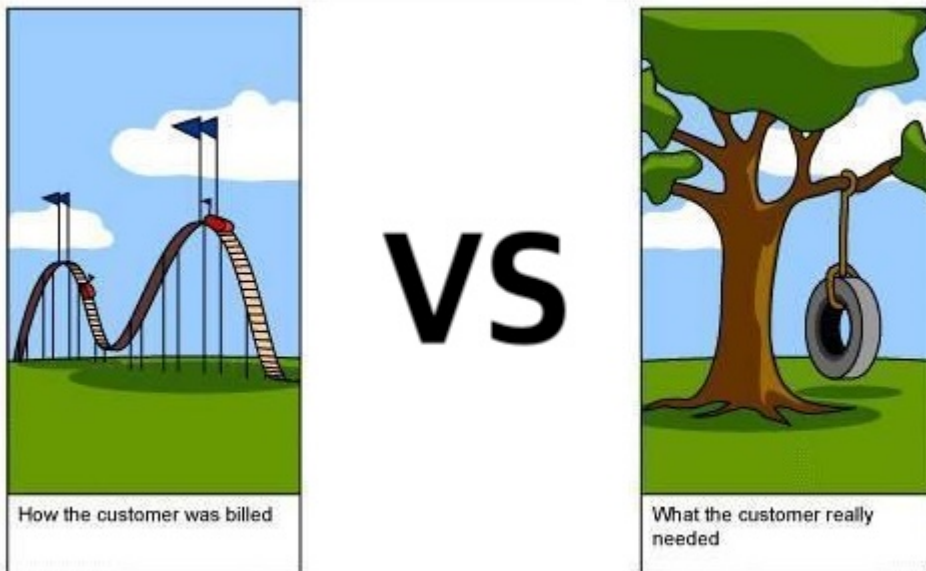
Vim을 오래 사용해왔지만 LSP를 쉽게 연결해주는 [coc.nvim](#)을 발견한 2018년 직후만큼 신나고 놀라웠던 순간은 드물었던 것 같다. Vim 플러그인을 제작하는 패러다임이 바뀌었기 때문이다.

- 접근성에 편의가 생겨 이전에 비해 훨씬 빨라진 속도로 수많은 플러그인이 쏟아져 나왔다.
 - 이전에는 필요한 것이 생기면 직접 만들어야 했는데 워낙 많은 것들이 쏟아져나오다 보니 찾아서 쓰기만 해도 되는 지점까지 순식간에 도달했다.
- 따라서 익숙한 Vim을 버리고 간헐적으로 사용하던 NeoVim으로 갈아타게 됐다.
 - 내 Vim에도 점점 더 많은 플러그인이 덧칠되었다.

👁️:늘어만 가는 복잡도

그러나 특정한 시점부터는(2021년 여름쯤?) 미묘하게 마음에 들지 않는 구석이 늘어나는 것을 느꼈다.

- 뭔가 점점 더 편해지긴 하는데 이상하게 지루하다. 재미가 없다.
- 뭐가 막 안 되어서 계속해서 머리를 써야 했던 **옛날 Vim이 훨씬 재밌었다.**
- 내가 직접 구성한 환경인데도 내가 파악하지 못하는 작동이 늘어나기 시작했다.
 - 계속 진행된다면 내가 파악하지 못하는 영역의 규모는 선형증가하다가 내 머리가 감당할 수 있는 한계선에서 진동하게 될 것이다.



내가 정말 원하는 것은 무엇인가?

- 내가 생각하는 Vim의 가장 큰 즐거움 중 하나는 통제감과 자유로움이다. 달려가는 것이 아니다.
 - IDE는 고치기가 어렵기 때문에 IDE로는 그런 감각을 얻기 어렵다.
 - IDE가 지하철이라면, Vim은 자전거 같은 느낌.

- 블랙박스가 점점 늘어난다면 내가 Vim을 사용해온 강력한 이유 하나가 흐려지는 것이다.

무엇보다도 다음과 같은 짜증나는 문제들이 일주일에도 두세번씩은 나를 귀찮게 했다.

- Vim에 추가해나가고 있는 수많은 의존관계를 직접 관리해야 하는데.. 그게 너무나 귀찮다.
- Vim 내에서 수행 가능한 무언가를 계속 추가한다면 복잡도는 그에 따라 계속해서 증가하게 된다.

2. 일반적 불확정성 원리

시스템은 성장하는 경향이 있다. 그리고 성장하는 대로 잠식한다. 바꾸어 말하면,

1. 복잡해진 시스템의 결과는 기대할 것이 없다.
2. 거대한 시스템의 전체적인 기능은 예상할 수 없다.

[발전형 : 시스템 기능 추가 불가능의 원리]

소형 시스템의 규모를 확대해서 만든 대형 시스템은 원래의 소형 시스템처럼 기능하지 않는다.

[1]

☺내가 Vim에 바라는 것들

특히 Vim을 사용해 풀타임으로 특정한 함수형 프로그래밍 언어를 사용하면서, 자기 자신의 필요에 대해 더 정교하게 정리해볼 수 있었다.

- 나의 Vim 사용에 있어 자동완성은 그닥 중요한 필요사항이 아니다.
 - 최소한만 되면 된다.
 - [Ultisnips](#) 스타일의 내가 직접 지정한 특수한 규칙들만으로도 차고 넘친다.
- 자동완성 기능이 더 많은 코드를 더 적은 키스트로크로 생성할수록, Vim의 NORMAL 명령을 사용하는 빈도는 줄어들게 된다.
 - 아이러니하게도 자동완성이 잘 될수록 Vim의 장점이 희석된다.
 - copilot 같은 코드 어시스트 도구가 더 많은 코드를 만들어줄수록 Vim을 쓰건 IDE를 쓰건 아무 상관이 없어질 것이다.

그리고 회사에서 업무를 하며 느낀 것들도 있었다.

- 결국 내가 작성한 코드들이 하는 대부분의 일은 데이터 변환이다.
- 코드도 데이터의 일종이다.
- 내가 하는 일의 대부분도 데이터 변환 작업인 것이다.
- 내가 바라는 Vim은 나의 데이터 변환 작업을 도와주는 도구여야 한다.
- 그런데 왜 IDE가 아니라 매번 Vim을 선택하려 하는가?
 - Vim의 NORMAL 명령이 텍스트를 처리할 때의 내 사고과정을 더 정밀하게 표현해주기 때문이다.
 - 표현일 뿐 아니라 실행이기도 하다. 이 대칭적 일치는 가볍게 넘길 수 없다.

· Vim은 그냥 하나의 UNIX 도구

이런 생각을 하다 보니 Vim 내에서 무엇을 하기보다는 Vim 바깥에서 Vim을 활용하는 방향으로의 작업을 많이 하게 되었다.

- 내가 회사에서 코딩을 제외하고 실제로 꽤 많이 하는 일은 터미널에서 명령어를 조합해 파이프라인을 만드는 일이다.
 - 그 다음은 그런 파이프라인을 사용해 csv 파일을 변환하는 일이다.
- 따라서 [sed](#) 같은 도구를 늘 사용하고 있기 때문에 정규식을 거의 매일 사용하고 있다.
 - 그러나 정규식의 복잡함 때문에 다른 사람들이 나와 똑같이 하지 못한다는 문제가 있음.
 - 가끔 정규식의 표현력에 **절차적 제약**을 추가하고 싶다는 생각이 들 때가 있다.

이러다보니 [sed](#)를 대체할 수 있으면서 Vim의 NORMAL 명령을 사용해보면 꽤 관찮을 수 있겠다는 생각이 들었다.

- 관점을 바꾼다. Vim에 플러그인을 얹지 말고, Vim을 perl이나 awk 같은 소모품 랭귀지로 사용한다.
- Vim 내부에 갇히지 말고 CLI를 더 자유롭게 사용하는 도구로 만들어보자.

vised

그래서 [vised](#) 라는 CLI 프로그램을 하나 만들어 보았다.

vi 명령을 사용할 수 있는 sed(stream editor)라는 뜻이다.

대충 다음과 같이 사용한다.

```
$ seq 101 111 | vided 'f6i !here! '  
101  
102  
103  
104  
105  
10 !here! 6  
107  
108  
109  
110  
111
```

```
$ seq 101 121 | xargs -n 3 | vided 'f a hello '  
101 hello 102 103  
104 hello 105 106  
107 hello 108 109  
110 hello 111 112  
113 hello 114 115  
116 hello 117 118  
119 hello 120 121
```

```
$ seq 101 121 | xargs -n 3 | vided 'f a hello ' '2flr-'  
101 hel-o 102 103  
104 hel-o 105 106  
107 hel-o 108 109  
110 hel-o 111 112  
113 hel-o 114 115  
116 hel-o 117 118  
119 hel-o 120 121
```

처음에는 perl로 만들었다가 코드를 알아보는 사람이 더 많아야 좋겠다는 생각이 들어서 자바스크립트로 다시 만들었다.

대단한 것은 아니고 `vim -es +normal +%print` 를 래핑하는 간단한 자바스크립트 프로그램이다. 작업 과정에서 [claude](#) 도움을 좀 받았다.

이 명령은 실제로도 터미널에서 사용이 가능한데, 자질구레한 옵션을 빠짐없이 입력해야 하고, stdin 입력을 받도록 하기가 성가시다.

```
$ seq 1 7 > test.txt  
  
$ vim -es '+%normal A hello' '+%print' ./test.txt  
1 hello  
2 hello  
3 hello  
4 hello  
5 hello
```

```
6 hello
7 hello
```

여기에는 자주 사용하고 있는 [moreutils](#)에 있는 [vipe 명령어](#), [vidir 명령어](#)가 좋은 영감을 줬다.

이 글을 읽고 Vim을 소모품처럼 사용하는 데에 관심을 갖게 된 분들이 있다면 [moreutils](#)의 두 명령도 함께 살펴보시는 것을 권한다.

· 주석

· 참고문헌

- 머피의 법칙 지혜의 패러독스 / 아더 블록 / 이인식 옮김 / 도서출판 까치 / 초판 5쇄 발행일 1994-12-10

1. 머피의 법칙 지혜의 패러독스. 시스템틱스 챕터. 136쪽. [↩](#)


반응 4개



댓글 0개 – Powered by [giscus](#)

작성하기 미리보기 Aa

댓글 작성하기



[> 로그아웃 댓글 등록