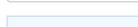
cgiseminar / curriculum

Branch: master ▼

curriculum / docs / command.md

Find file

Copy path



🍆 kim hanwoong 내용수정함.

d69b251 on 8 Jan

0 contributors

52 lines (40 sloc) | 2.75 KB

Command

유닉스, 리눅스가 지금까지 죽지 않고 살아남는 것은 우연이 아닙니다. 오늘은 몇가지 특징중에서 명령어를 구성하고 있는 특징들을 알아보겠습니다. 리눅스에는 Is라는 명령어가 있습니다. Unix와 인생을 함께한 명령어이니 1980~1988 사이에 Is명령의 주요 기능들은 완성되었을 것 입니다. 어림잡아도 30살이 넘은 명령어입니다. 영상 제작에 자주 사용되는 ffmpeg 라는 명령어도 역사상 약 18년정도로 오래된 명령어 입니다.

명령어의 구성

모든 리눅스 명령어는 아래 구조를 지닙니다. 인수는 영어로 줄여서 argv 라고도 불립니다. 잘 기억해 두세요.

```
$ 명령어 인수...
```

조금 더 디테일하게 정의하면 아래 형태를 띄기도 합니다.(프로그래머의 성향상 아닌경우도 있습니다.)

```
$ 명령어이름 옵션A 값 옵션B 값 옵션C 값 ...
```

우리는 그 전까지 실습을 통해서 수많은 명령어를 사용했지만, 이 구조가 얼마나 효율적인지 고민해본 적은 없습니다. 인수에 대해서 우리는 프로 그래밍을 설계할 때 하나의 명령어에 대해서 직교적인 설계를 해야합니다.

```
하나의 명령어에 대해서...
옵션1 = 값,
옵션2 = 값,
옵션3 = 값,
```

여러분이 앞으로 프로그래밍을 통해서 명령어를 생성할 때, 위 특징을 가지고 제작한다면 이미 훌륭한 설계이며 여러분이 만든 명령어가 오랫동 안 살아남을 확률을 만들고 있다는 증거입니다. 옵션은 아주 세부적으로 나누어 놓고 각 옵션이 서로에게 영향을 주지 않도록 설계합니다. 잘 구 성된 인수는 추후에 GUI을 연결할 때에도 편하게 적용할 수 있습니다.

시간이 흘러서 GUI를 만들더라도 각 옵션에 해당하는 각각의 GUI만 컴포넌트만 만들어 주고 상호간에 옵션만 연결하면 되기 때문입니다.

심플한 명령어를 만드는 단계

- 1. 명령어 이름을 짓는다. 가장 중요합니다.
- 2. 명령어에 필요한 옵션을 만든다.(자주 사용하는 값은 디폴트 값을 설정해둡니다.)
- 3. 프로그래밍 한다.
- 4. 작동이 잘 되는지 테스트한다.
- 5. 필요한 옵션이 있다면 직교적으로 늘린다.

이 공식만 따라서 제작하더라도 이미 수십년간 Unix, Linux에서 살아남은 유명한 명령어의 큰 설계 개념을 따라하게 되는 것 입니다.

또한 이 방식은 여러분이 프로그래밍에서 함수를 만들 때와도 굉장히 닮아있습니다.

```
def 함수명(인수=값, 인수=값, 인수=값):
함수내용...
```