



# ArgParse

Multi Language Command Line Parser Creator

Diogo Eugênio  
Willie Lawrence



Imagine que você precisa criar um programa em C++ que recebe alguns argumentos na linha de comando

```
Search Terminal Help
seLang$ cat_simulator -w
```

```
1  void walk() {
2      ...
3  }
4
5  int main(int argc, char const *argv[]) {
6      if (w) {
7          walk()
8      }
9      return 0;
10 }
```

# Para implementar um parser em C++ hoje

```
1  #include <stdio.h>      /* for printf */
2  #include <stdlib.h>     /* for exit */
3  #include <unistd.h>     /* for getopt */
4  int main (int argc, char **argv) {
5      int w = 0;
6      char *copt = 0, *dopt = 0;
7      while ((c = getopt(argc, argv, "w")) != -1) {
8          int this_option_optind = optind ? optind : 1;
9          switch (c) {
10             case 'w':
11                 w = 1;
12                 break;
13             case '?':
14                 break;
15             default:
16                 printf ("?? getopt returned character code 0%o ??\n", c);
17             }
18         }
19         if (optind < argc) {
20             printf ("non-option ARGV-elements: ");
21             while (optind < argc)
22                 printf ("%s ", argv[optind++]);
23             printf ("\n");
24         }
25         exit (0);
26     }
```

# Para implementar um parser em bash hoje

```
1  BG=unset
2  F=unset
3  NUMBERS=unset
4  PARSED_ARGUMENTS=$(getopt -a -n $0 -o "w" -- "$@")
5
6  VALID_ARGUMENTS=$?
7  if [ "$VALID_ARGUMENTS" != "0" ]; then
8      print_usage
9  fi
10
11  eval set -- "$PARSED_ARGUMENTS"
12  while :
13  do
14      case "$1" in
15          -w ) F=1; shift ;;
16          --) shift; break ;;
17          *) print_usage ;;
18      esac
19  done
20  NUMBERS=$@
21
```

# Para implementar um parser em ArgParse hoje

```
1 Este programa simula um gato
2
3
4 FLAG: w Faz o gato andar
5 FLAG: m Faz o gato miar|
```

# **ArgParse objetiva especificar/implementar muito mais rapidamente que argumentos um programa recebe**

- Gera como saída um código na linguagem desejada que faz o trabalho de parse da linha de comando e entrega de forma mais fácil pro programador uma forma de saber se um determinado argumento foi passado ou não e qual o valor
- Suporta atualmente duas linguagens de saída:
  - Python 2/3
  - Bash

# Exemplo com todas as funcionalidades

Descrição do meu programa

FLAG: f          Descrição da minha flag

ATTR: name      Descricao do atributo

POS: foto        Descrição do atributo posicional

ARGS: fotos     Descrição do atributo com quantidade indefinida de items

# **Tipo de argumento de linha de comando**



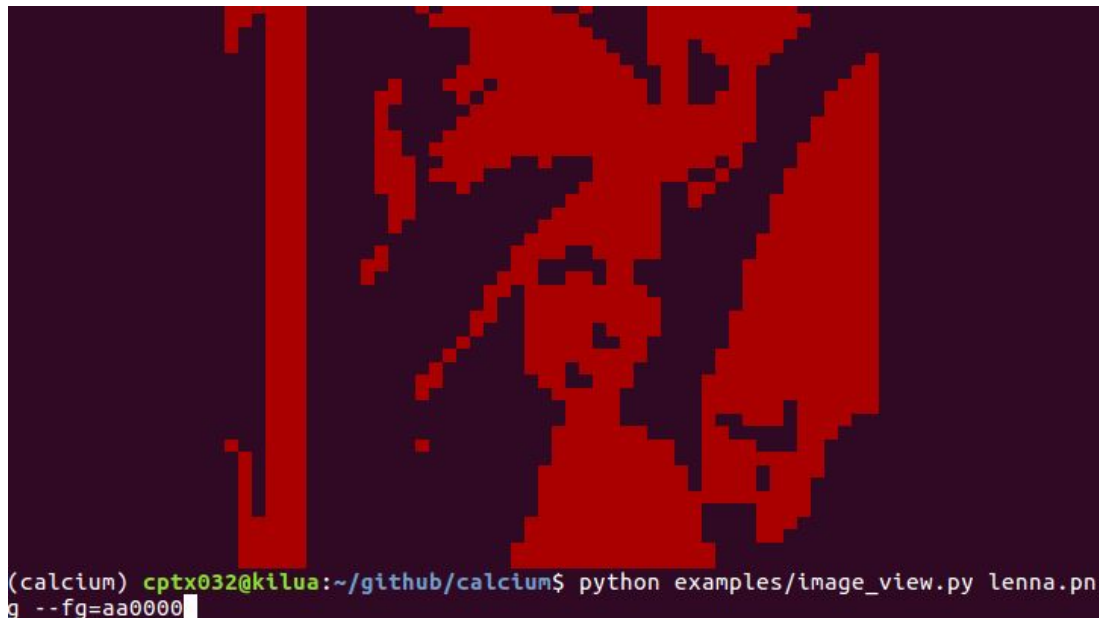
# Flag

Um argumento da linha de comando que não precisa de valores adicionais, a simples presença dele nos argumentos já é suficiente. Geralmente é usada para indicar atributos booleanos



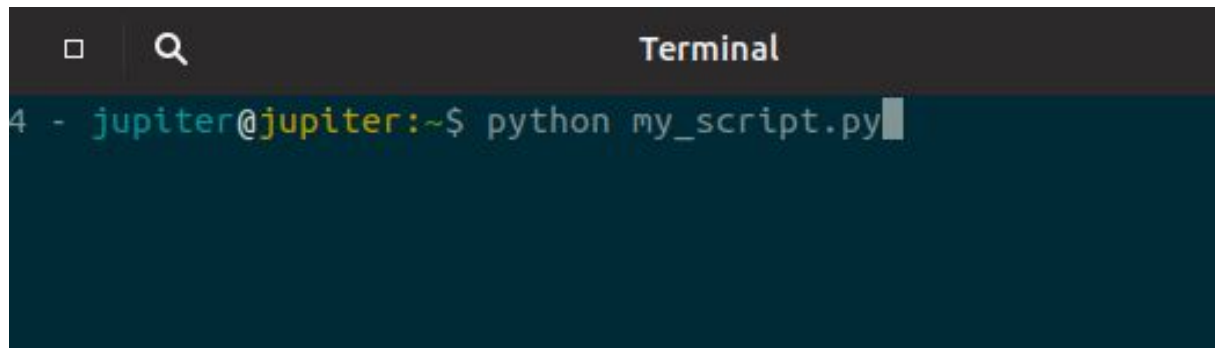
# Named Attribute

Um argumento da linha de comando que precisa de valores adicionais.



# Positional Argument

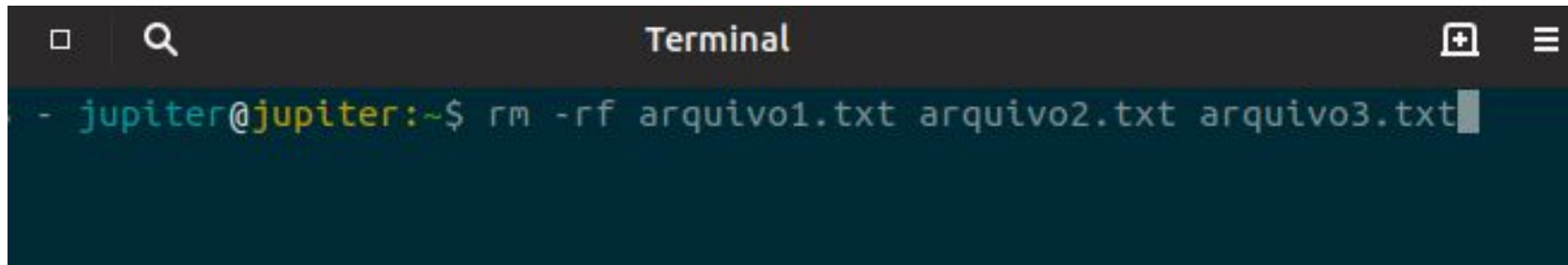
Um único argumento da linha de comando que não precisa ter nenhum nome associado. Se a flag só precisa do nome e não do valor, o positional argument só precisa do valor, e não do nome.

A terminal window with a dark background. The title bar at the top is dark gray and contains a square icon, a magnifying glass icon, and the word "Terminal". The terminal content shows a prompt "4 - jupyter@jupyter:~\$" followed by the command "python my\_script.py" and a white cursor at the end.

```
4 - jupyter@jupyter:~$ python my_script.py
```

# Multi value positional arguments

Argumentos da linha de comando que não precisam ter nenhum nome associado. Se a flag só precisa do nome e não do valor, o multi value positional argument só precisa dos valores.

A terminal window titled "Terminal" with a dark background. The prompt is "jupiter@jupiter:~\$". The command entered is "rm -rf arquivo1.txt arquivo2.txt arquivo3.txt".

```
Terminal  
- jupiter@jupiter:~$ rm -rf arquivo1.txt arquivo2.txt arquivo3.txt
```

# Exemplo Prático



# Obrigado!

