乱弹: CLI 中的编译原理

墨泪

2020-03-29

大纲

- iRedis 中的输入流解析
- 编译原理是什么
- iRedis 中的编译原理
- Q&A

iRedis の 输入流

核心疑问: iRedis 读取输入后, 是怎么进行后续处理的?

```
iredis  1.5.0 (Python 3.7.6)
redis-server  5.0.7
Home:   https://iredis.io
Issues: https://iredis.io/issues
127.0.0.1:6379> set foo bar xx
(nil)
127.0.0.1:6379> get foo
(nil)
127.0.0.1:6379> set foo bar nx
OK
127.0.0.1:6379> get foo
"bar"
```

test_utils.py

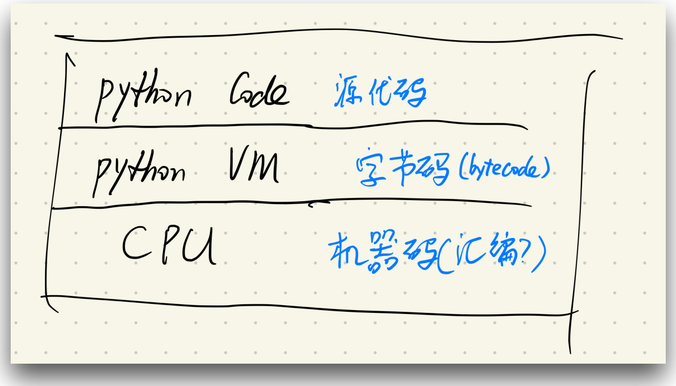
tests > unittests > test_utils.py

```python
@pytest.mark.parametrize(
    "test_input,expected",
    [
        ("hello world", ["hello", "world"]),
        ("'hello world'", ["hello world"]),
        ('''hello"world"''', ["helloworld"]),
        (r'''hello\"world"''', [r"hello\world"]),
        (r'"\\"', [r"\\"]),
        (r"\\", [r"\\"]),
        (r"\abcd ef", [r"\abcd", "ef"]),
        # quotes in quotes
        (r""" 'hello"world' """, ['hello"world']),
        (r""" "hello'world" """, ["hello'world"]),
        (r""" 'hello\'world'""", ["hello'world"]),
        (r""" "hello\"world" """, ['hello"world']),
        (r"''", [""]),  # set foo "" is a legal command
        (r'""', [""]),  # set foo "" is a legal command
        (r"\\", ["\\\\"]),  # blackslash are legal
        ("\\hello\\", ["\\hello\\"]),  # blackslash are legal
    ],
)
def test_stipe_quote_escaple_in_quote(test_input, expected):
    assert list(strip_quote_args(test_input)) == expected
```

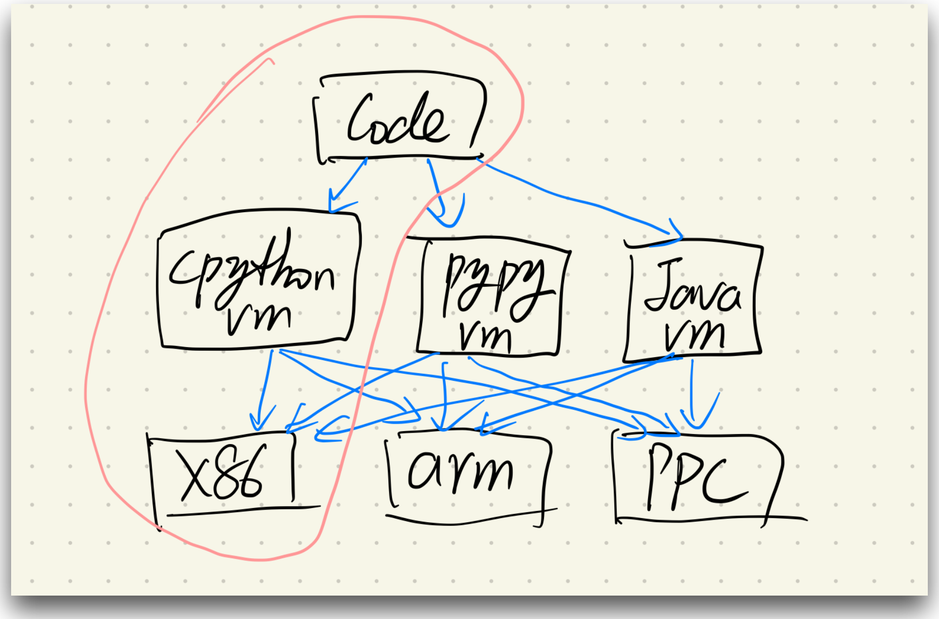
```
Case Num                    Input -> Output                    -> Expected                Status
Case  0:              "hello world" -> ["hello", "world"]      -> ["hello", "world"]      Success
Case  1:            "'hello world'" -> ["hello world"]         -> ["hello world"]         Success
Case  2:          "hello\"world\"" -> ["helloworld"]           -> ["helloworld"]          Success
Case  3:         "hello\\\"world\"" -> ["hello\\world"]        -> ["hello\\world"]        Success
Case  4:               "\"\\\\\"" -> ["\\\\"]                 -> ["\\\\"]                Success
Case  5:                 "\\\\" -> ["\\\\"]                   -> ["\\\\"]                Success
Case  6:             "\\abcd ef" -> ["\\abcd", "ef"]          -> ["\\abcd", "ef"]        Success
Case  7:       " 'hello\"world' " -> ["hello\"world"]         -> ["hello\"world"]        Success
Case  8:     " \"hello'world\" " -> ["hello'world"]           -> ["hello'world"]         Success
Case  9:      " 'hello\\'world'" -> ["hello'world"]           -> ["hello'world"]         Success
Case 10:  " \"hello\\\"world\" " -> ["hello\"world"]          -> ["hello\"world"]        Success
Case 11:                   "''" -> [""]                       -> [""]                    Success
Case 12:                 "\"\"" -> [""]                       -> [""]                    Success
Case 13:                 "\\\\" -> ["\\\\"]                   -> ["\\\\"]                Success
Case 14:           "\\hello\\" -> ["\\hello\\"]               -> ["\\hello\\"]           Success
```

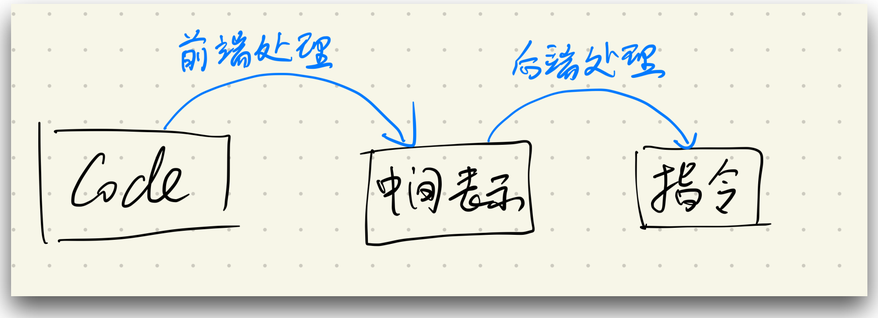
编译原理简介

- 编译原理: 是一种”翻译”过程的系统化抽象原理



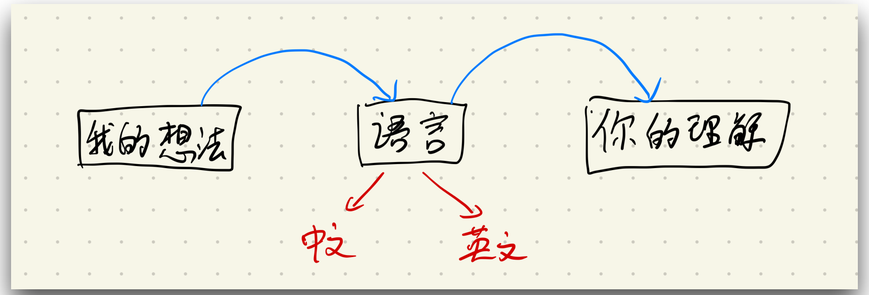
Python 语言编译架构



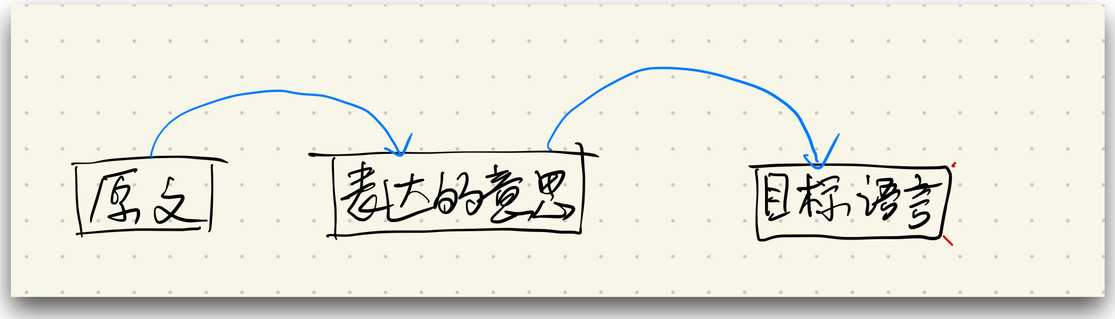


编译原理抽象过程

生活中 の 编译原理



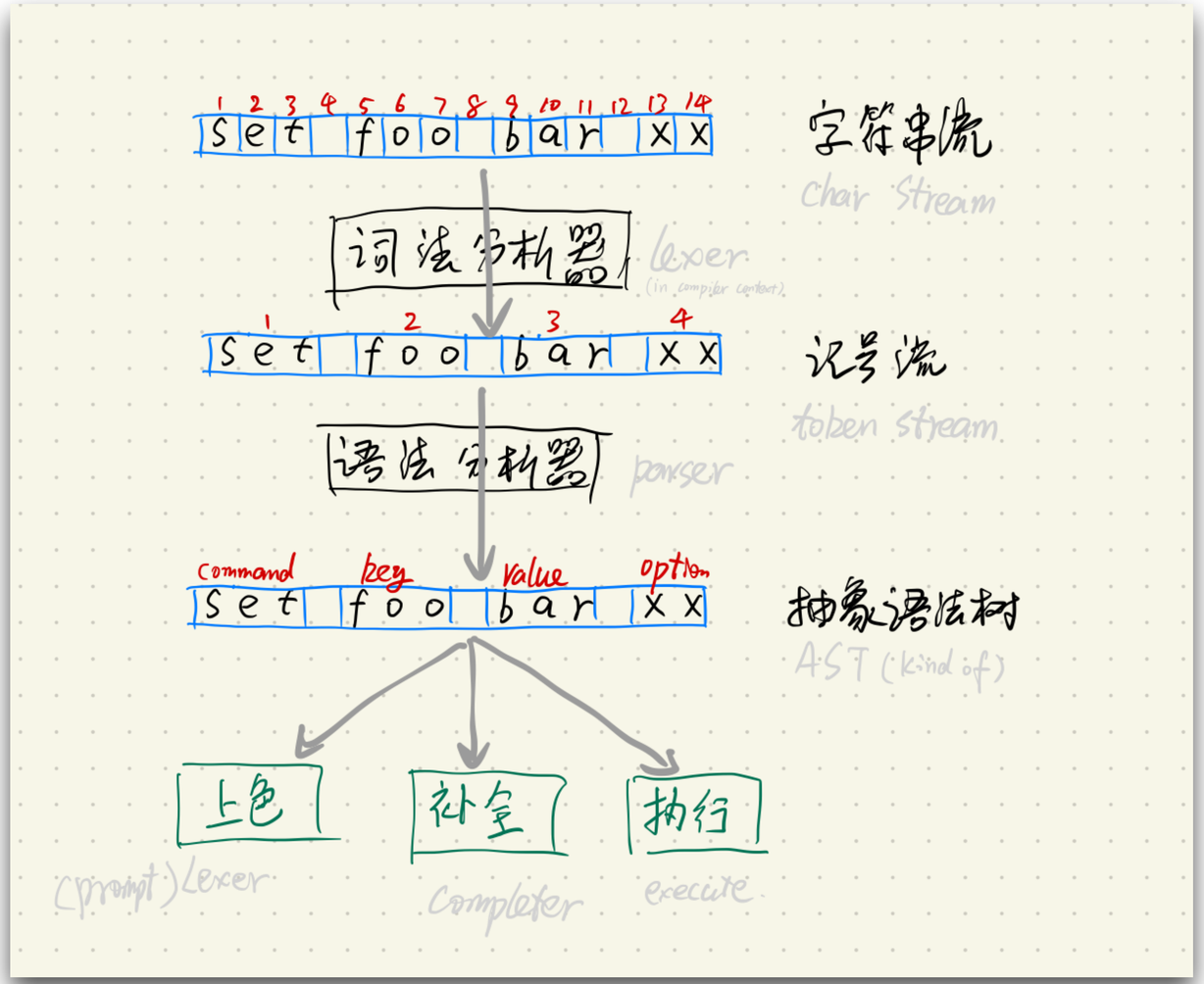
人和人之间的沟通交流



语言和语言之间的翻译

iRedis 中的编译原理

```
iredis  1.5.0 (Python 3.7.6)
redis-server  5.0.7
Home:   https://iredis.io
Issues: https://iredis.io/issues
127.0.0.1:6379> set foo bar xx
(nil)
127.0.0.1:6379> get foo
(nil)
127.0.0.1:6379> set foo bar nx
OK
127.0.0.1:6379> get foo
"bar"
```



iRedis 命令解析实现


```python
def strip_quote_args(s):
    """
    Given string s, split it into args.(Like bash paring)
    Handle with all quote cases.

    Raise ``InvalidArguments`` if quotes not match

    :return: args list.
    """
    word = []
    in_quote = None
    pre_back_slash = False
    for char in s:
        if in_quote:
            # close quote
            if char == in_quote:
                if not pre_back_slash:
                    yield "".join(word)
                    word = []
                    in_quote = None
                else:
                    # previous char is \, merge with current "
                    word[-1] = char
            else:
                word.append(char)
        # not in quote
        else:
            # sperator
            if sperator.match(char):
                if word:
                    yield "".join(word)
                    word = []
            # open quotes
            elif char in ["'", '"']:
                in_quote = char
            else:
                word.append(char)
        if char == "\\" and not pre_back_slash:
            pre_back_slash = True
        else:
            pre_back_slash = False

    if word:
        yield "".join(word)
    # quote not close
    if in_quote:
        raise InvalidArguments("Invalid argument(s)")
```

- 按字符读入缓冲区
- 空格在 ‘in_quote’ 状态机下的处理
- 转义符在各种状态下的处理
- 最终完成了记号流的输出

为什么不用 Shlex

- 场景不同: iRedis 在每一个字符键入时进行解析
- 标准不同: redis-cli 的解析标准和 bash 不一致

```
>>> shlex.split("set foo 'bar ")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/System/Library/Frameworks/Python
    return list(lex)
  File "/System/Library/Frameworks/Python
    token = self.get_token()
  File "/System/Library/Frameworks/Python
    raw = self.read_token()
  File "/System/Library/Frameworks/Python
    raise ValueError, "No closing quotati
ValueError: No closing quotation
```

不允许未关闭的引号字符

```
[ruohan.chen@Ruohans-15Inch-MBP /Users/ruohan.chen]
$ redis-cli
127.0.0.1:6379> set key \hello\
OK
127.0.0.1:6379> get key
"\\hello\\"
127.0.0.1:6379>

ValueError: No closing quotation
>>> shlex.split("\\hello\\")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/System/Library/Frameworks/Python.framework/Versions/2.7
    return list(lex)
  File "/System/Library/Frameworks/Python.framework/Versions/2.7
    token = self.get_token()
  File "/System/Library/Frameworks/Python.framework/Versions/2.7
    raw = self.read_token()
  File "/System/Library/Frameworks/Python.framework/Versions/2.7
    raise ValueError, "No escaped character"
ValueError: No escaped character
```

不允许转义符号后为空

```
127.0.0.1:6379> set key \abc
OK
127.0.0.1:6379> get key
"\\abc"
127.0.0.1:6379>

>>> shlex.split(r"\abc")
['abc']
>>>
```


无意义转义时的理解不同

演示环节

- 介绍 SLY(Sly Lex Yacc)
- 基于 SLY 写的 iRedis “编译器” 走读

Q&A