

---

# Fong Documentation

*Release 1.0*

**fong**

May 24, 2018



## 目录

|   |             |    |
|---|-------------|----|
| 1 | C/C++       | 3  |
| 2 | Python      | 5  |
| 3 | Linux/Shell | 7  |
| 4 | 机器学习        | 9  |
| 5 | 深度学习        | 11 |
| 6 | 资源链接        | 13 |
| 7 | 实用软件        | 15 |
| 8 | 其他          | 17 |



---

Note: 文中可能存在错误，欢迎 PR。

---



## 1.1 main 函数

### 1.1.1 返回值

C++ main 函数的返回值必须是 `int`，即整型类型。在大多数系统中，main 的返回值被用来指示状态，返回值 0 表示执行成功，非 0 的返回值含义由系统定义，通常用来指出错误类型。

Windows 系统下运行可执行文件（如 `launch.exe`）可以直接忽略其扩展名 `.exe`：

```
launch
```

Unix 系统下需要使用全文件名，包括扩展名：

```
./a.out
```

访问 main 函数返回之后的方法依赖于系统。在 Windows 和 Unix 系统中，执行完一个程序之后，都可以通过 `echo` 命令来获取返回值。

Windows:

```
echo %ERRORLEVEL%
```

Unix:

```
echo $?
```

### 1.1.2 处理命令行选项

main 函数的形参列表有两种形式：

```
int main(int argc, char *argv[]){ ... }

int main(int argc, char **argv){ ... }
```

第一种形参 `*argv[]` 中，`argv` 是一个数组，它的元素是指向 C 风格的字符串的指针；第二种形参 `**argv` 中，`argv` 指向 `char*`。参数 `argc` 表示数组中字符串的数量。

当实参传给 `main` 函数之后，`argv` 的第一个元素指向程序的名字或者一个空字符串，接下来的元素依次传递命令行提供的实参。最后一个指针之后的元素值保证为 0。例如，执行：

```
launch -d -o ofile data
```

`launch` 是可执行文件。那么，`argc=5`，`argv` 包含如下的 C 风格字符串：

```
1 argv[0] = "launch";
2 argv[1] = "-d";
3 argv[2] = "-o";
4 argv[3] = "ofile";
5 argv[4] = "data";
6 argv[5] = "0";
```

---

**Note:** 当使用 `argv` 中的实参时，实参是从 `argv[1]` 开始的；`argv[0]` 保存的是程序名，而非用户输入。

---

### 1.1.3 参考资料

《C++ Primer 第 5 版中文版》Page 2, Page 197。



---

CHAPTER

TWO

---

PYTHON



---

CHAPTER

THREE

---

LINUX/SHELL



---

CHAPTER

FOUR

---

机器学习









## 资源链接

### 6.1 Github Page

<https://fongyk.github.io/>

### 6.2 arXiv

<https://arxiv.org/>

### 6.3 Read the Docs

<https://readthedocs.org/>

### 6.4 C++ Reference

<http://www.cplusplus.com/reference/>

### 6.5 Numpy

<http://cs231n.github.io/python-numpy-tutorial/>

### 6.6 Pytorch

Tutorials: <https://pytorch.org/tutorials/>

Docs: <https://pytorch.org/docs/master/index.html>

## 6.7 Stanford University Lectures

CS229: <http://cs229.stanford.edu/syllabus.html>

CS231: <http://cs231n.github.io/>

## 6.8 ShareLatex

<https://www.sharelatex.com/login>

## 6.9 PlanetB

<http://www.planetb.ca/syntax-highlight-word>

## 6.10 Vision Open Source Library

检索: <http://yael.gforge.inria.fr/index.html>

特征: <http://www.vlfeat.org/index.html>

## 6.11 牛客网

<https://www.nowcoder.com/>

## 实用软件

### 7.1 Listary

---

**Note:** Windows 下快速查找文件及应用程序

---

<http://www.listary.com/>

### 7.2 FreeCommander

---

**Note:** Windows 下的资源管理器

---

<https://freecommander.com/en/summary/>

### 7.3 MobaXterm

---

**Note:** Windows 下连接服务器的终端

---

<https://mobaxterm.mobatek.net/>

### 7.4 TeamViewer

---

**Note:** 远程连接

---

<https://www.teamviewer.com/zhCN/>

## 7.5 Notepad++

---

**Note:** 强大的文本阅读/编辑器

---

<https://notepad-plus-plus.org/>

## 8.1 rst 语法测试

makefile 规则:

```
target ... : prerequisites ...  
    command  
    ...  
    ...
```

下面是几个定义:

**target** 可以是一个 object file (目标文件), 也可以是一个执行文件, 还可以是一个标签 (label)。对于标签这种特性, 在后续的“伪目标”章节中会有叙述。

**prerequisites** 生成该 target 所依赖的文件和/或 target

**command** 该 target 要执行的命令 (任意的 shell 命令)

这是一个文件的依赖关系, 也就是说, target 这一个或多个的目标文件依赖于 prerequisites 中的文件, 其生成规则定义在 command 中。说白一点就是说:

prerequisites 中如果有一个以上的文件比 target 文件要新的话, command 所定义的命令就会被执行。

这就是 makefile 的规则, 也就是 makefile 中最核心的内容。

echo "Hello World!";

行内公式使用 math 这个 role:  $a^2 + b^2 = c^2$ .

$$\begin{aligned}(a + b)^2 &= (a + b)(a + b) \\ &= a^2 + 2ab + b^2\end{aligned}$$

latex math 测试:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \dots, N-1.$$

将高亮语言设置为 C

测试 C

```
1 int a = 0;
2 char c = 'c';
3 printf("%c\n", c);
```

这里是 C++ :

```
1 int main()
2 {
3     int i;
4     int j;
5     cin >> i >> j;
6     cout << i << j << endl;
7     return 1;
8 }
9 // 主函数注释
```

斜体 *text*

将高亮语言设置为 python

测试 python

```
1 import torch
2 import numpy as np
3 print "hello world"
```

这里是 python (code):

```
1 def foo():
2     print "Love Python, Love FreeDome"
3     print "E 文标点,.0123456789, 中文标点,. "
```

如果数据库有问题, 执行下面的 SQL:

```
-- Dumping data for table `item_table`
INSERT INTO item_table VALUES (
0000000001, 0, 'Manual', '', '0.18.0',
'This is the manual for Mantis version 0.18.0.\r\n\r\nThe Mantis manual is modeled after the
↪[url=http://www.php.net/manual/en/]PHP Manual[url]. It is authored via the "\"manual\" module
↪in Mantis CVS. You can always view/download the latest version of this manual from [url=http://
↪mantisbt.sourceforge.net/manual/]here[url].',
'', 1, 1, 20030811192655);
```

下面是 python:

```
1 # 测试注释
2 def foo():
3     print "Love Python, Love FreeDome"
4     print "E 文标点,.0123456789, 中文标点,. "
```

下面是 javascript 的 rst 源码:

```
1 .. code-block:: javascript
2     :linenos:
3
4     function whatever()
5     {
6         return "such color"
7     }
```

下面是 python (code-block):

```
1 def foo():
2     print "Love Python, Love FreeDome"
3     print "E 文标点,.0123456789, 中文标点,. "
```

下面是 bash :

```
1 cd home
2 echo $PATH
3 source ~/.bashrc
4 ls -l
5 mkdir filefolder
6 cd ..
```

结束