

# 导言

这份文档主要用来存放一些实际工作中碰到的实用的代码片段，可能包含 MATLAB、Python、C 和一些  $\text{\LaTeX}$  的小知识。由于我是一个化学工程的学生，同时也初涉编程未深，计算机专业知识难免会出点错误，欢迎指正。

如果有人想编译这份手册或想学习一下实现，请务必读以下说明。

字体设置，为了避免侵权，尽可能使用开源字体<sup>①</sup>。

- Source Han Sans: <https://github.com/adobe-fonts/source-han-sans/tree/release>
- Source Han Serif: <https://github.com/adobe-fonts/source-han-serif/tree/release>
- Source Code Pro: <https://github.com/adobe-fonts/source-code-pro>
- PT Sans Narrow: <https://fonts.google.com/specimen/PT+Sans+Narrow>
- TeX Gyre: 有问题前往<https://www.ctan.org>获取，一般来说 TeX 发行版自带
- 等宽字体：大多数等宽字体都是程序员使用的，开源居多，颇易获取。我常用 DejaVu Sans Mono, Fira Code 和 Source Code Pro 三种。

%% 字体设置

```
\usepackage{fontspec}
\setmainfont{Adobe Garamond Pro} % TeX Gyre Pagella
\setsansfont{TeX Gyre Heros}
\setmonofont{Source Code Pro} % Consolas, DejaVu Sans Mono
\setCJKmainfont[BoldFont={Source Han Sans SC},
  ↳ ItalicFont={KaiTi}]{Source Han Serif SC}
\setCJKmonofont{FangSong}
\setCJKsansfont{Source Han Sans SC}
```

%% 数学字体

```
\usepackage{unicode-math}
\setmathfont[math-style = ISO, bold-style = ISO]{TeX Gyre Pagella Math}
```

%% url 样式

```
\newfontfamily\urlfont{PT Sans Narrow}
```

编译环境设置，代码高亮环境由 minted 宏包提供（需要 Python 环境）。

代码测试环境，各种代码的运行环境为 MATLAB 2017a、Anaconda 4.4.0（Python 3.6x）、Visual Studio 2017 community、MiKTeX 2.9（各宏包均为最新）。

---

<sup>①</sup> 西文主字体 Adobe Garamond Pro、楷体、仿宋暂时没有找到理想的替代方案

# 第一章 MATLAB

## 问题 1 如何遍历当前文件夹及其子文件夹中的全部文件

假设现在我们有这样一个文件夹 A，它含有一些文件和子文件夹 B、C、D.....，这些子文件夹又包含若干层子文件夹。我们需要将这个父文件夹（A）及其子文件夹（B、C、D.....）和孙文件夹中的所有文件名和其路径取出来。

如果你用的是 MATLAB 2016b 及更新的版本，那真的太棒了！`dir()` 函数已经支持遍历搜索了。尝试敲入：

```
dir_data = dir('*/*');  
dir_data([dir_data.isdir]) = []; % 去除所有 . 和 .. 文件夹
```

这将会返回一个包含文件信息的 struct，现在你可以任意操作这些 struct 了，随意拼接路径。解放大脑，哦也！方便归方便，但是，一来肯定有大多数人使用的是 MATLAB 2016b 之前的版本，二来，解放大脑意味着我失去了一次独立思考的机会。

### 思考

对于实现方法<sup>①</sup>，多层次的遍历，我第一时间想到的是递归。然后就是数据的存储了，`dir()` 函数返回的是一个 struct，这个数据结构储存有文件的信息，我们要充分利用这个数据结构。所以现在思路是，写一个递归函数，这个函数返回包含所有文件信息的 struct。

这个函数应对先处理父文件夹，获取文件和子文件夹，然后储存文件信息，同时去除子文件夹中的 '.' 和 '..' 这两个特殊文件夹。我们对获取的子文件夹再次调用该函数，并储存文件信息。如此，利用递归获取子子孙孙无穷尽文件夹的信息<sup>②</sup>，最后函数返回存储有所有文件信息的 struct。现在，你可以对这个结构体做你想做的事情。思路如算法 1.1 所示。

### 解

MATLAB 2016b 以上的版本我们可以用函数返回 struct，这个数据结构包含 [folder, name, date, bytes, isdir, datenum] 六个字段的信息，我们可以按自己意愿使用 folder 和 name 拼接出文件的完整路径。

```
% get all file name in current dir and sub dir, Compatible with MATLAB  
↳ R2016b and newer  
function file_list = get_all_file_name_R2016b_newer(path)  
  
dir_data = dir(path);
```

<sup>①</sup> 思路来源：How to get all files under a specific directory in MATLAB?

<sup>②</sup> 其实这并不可能，因为递归是有栈高度限制的，调用函数压入栈，返回函数弹出栈，如果文件夹层次太深，一直压栈就会到达栈溢出警告的极限，例如 Python 的栈往往是 100 层，我想 MATLAB 的栈也大致如此，不会太高

**Algorithm 1.1:** 遍历获取当前文件夹及其所有子文件夹中的文件名

---

```

Function get_all_file_name(path):
    get file and sub_dir information of current dir;
    storing file information;
    remove specific folder of sub_dir;
    for first sub_dir to last sub_dir do
        get next sub_dir;
        recursion  $\Rightarrow$  get_all_file_name(path);
    end
    return struct of file information;
end

```

---

```

file_list = dir_data(~[dir_data.isdir]); % file name of current dir

% get sub dir information
sub_dir = dir_data([dir_data.isdir]); % struct
dot_dir = ismember({sub_dir.name}, {'.', '..'}); % logical
sub_dir = sub_dir(~dot_dir); % struct, remove specific folder

% recursion
for i = 1:length(sub_dir)
    next_dir = fullfile(sub_dir(i).folder, sub_dir(i).name); % str
    file_list = [file_list; get_all_file_name_R2016b_newer(next_dir)]; %
    % struct
end

end

```

MATLAB 2016a 及之前的版本 dir struct 信息并不包含 folder, 如果返回 struct, 将只有文件的 [name, date, bytes, isdir, datenum] 五个字段的信息, 所以我们并不能根据函数返回的 struct 拼接出文件完整路径, 我们需要自己将路径拼接成一个 cell, 然后使用函数返回 cell。

```

% get all file name in current dir and sub dir, Compatible with MATLAB
% R2016a and older
function file_list = get_all_file_name_R2016a_older(path)

% file name of current dir
dir_data = dir(path);
file_list_struct = dir_data(~[dir_data.isdir]);
file_list = fullfile(path, {file_list_struct.name})';

% get sub dir information

```

```
sub_dir = dir_data([dir_data.isdir]); % struct
dot_dir = ismember({sub_dir.name}, {'.', '..'}); % logical
sub_dir = sub_dir(~dot_dir); % struct, remove specific folder

% recursion
for i = 1:length(sub_dir)
    next_dir = fullfile(path, sub_dir(i).name); % str
    file_list = [file_list; get_all_file_name_R2016a_older(next_dir)]; %
    ~ struct
end
end
```

## 总结

`dir()` 函数遍历整个 F 盘共 2 万余文件文件大约需要 1.555823s。我们实现的递归函数遍历 F 盘文件大约需要 3.703009s。慢是慢了点，但我们成功运用了递归解决问题，不是吗？

## 问题 2 title 中英文标题

## 第二章 Python

### 问题 1 如何展开一个嵌套的序列？

我们现在有这样一个序列 `items = [1, 2, [3, 4, [5, 6, [9, 8], 7], 8]]`，我们想逐级展开这个序列，然后将所有元素装入一个序列。

如果这个序列层级较少，我们可以用多层 `for` 循环来遍历这个序列。一旦这个序列超过 3 层，过多的循环会让你很头疼。同样，这种多层级的问题我们可以用递归来解决。构建一个函数，这个函数能处理第一层的元素，由于第二层是 `list`，它是一个可迭代对象，我们只需要判断第二层是不是可迭代对象，同时忽略 `str`，`bytes` 对象<sup>①</sup>。只要内层是可迭代的，我们就开始递归，对其应用该函数。

```
from collections import Iterable

def unfold(items, unfolded=None, ignore_types=(str, bytes)):
    unfolded = list() if unfolded is None else unfolded

    for item in items:
        if isinstance(item, Iterable) and not isinstance(item,
            ignore_types):
            unfold(item, unfolded=unfolded)
        else:
            unfolded.append(item)

    return unfolded
```

由于存在递归，所以函数会被调用很多次，每次调用所得的数据都需要保留，如何在多次的调用之间共享保留数据呢？我采用一个默认参数来实现，首次调用时不给默认参数新值，这会产生一个空的 `list`，当对内层对象调用时，将上一次产生的数据赋值给这个参数。输出结果：

```
>>> items1 = ['Paula', ['Thomas', 'Lewis', ['siyu', 'ziyan', ['jianyuan']]]]
>>> items2 = [1, 2, [3, 4, [5, 6, [9, 8], 7], 8]]
>>> items3 = [[1, 2], 3, (4, [5, 6])]
>>> print(unfold(items1))
>>> print(unfold(items2))
>>> print(unfold(items3))
```

<sup>①</sup> `str`，`bytes` 也是可迭代对象，我们要避免其展开成单个字符。

```
['Paula', 'Thomas', 'Lewis', 'siyu', 'ziyan', 'jianyuan']  
[1, 2, 3, 4, 5, 6, 9, 8, 7, 8]  
[1, 2, 3, 4, 5, 6]
```

但这样做有两个显而易见的坏处，一是当我们的嵌套序列有无限多层，递归会栈溢出；二是序列整个被读取到内存中了，当序列元素非常多，比如 1 亿，内存会被撑死。坏处一我们不去管他，大多数情况下是适用的，坏处二可以很容易的利用 generator 来解决<sup>②</sup>。

```
from collections import Iterable  
  
def unfold(items, ignore_types=(str, bytes)):  
  
    for item in items:  
        if isinstance(item, Iterable) and not isinstance(item,  
            ~ ignore_types):  
            yield from unfold(item)  
        else:  
            yield item
```

使用 generator 一来能防止内存爆炸，二来不需要在函数的多次调用见传递数据，代码更清晰明朗。需要注意，generator 是惰性序列，边调用边计算，我们需要使用 **for** 迭代出每一个元素或者直接用 **list()** 获取全部元素。

```
items1 = ['Paula', ['Thomas', 'Lewis', ['siyu', 'ziyan', ['jianyuan']]]]  
items2 = [1, 2, [3, 4, [5, 6, [9, 8], 7], 8]]  
items3 = [[1, 2], 3, (4, [5, 6])]  
print(list(unfold(items1)))  
print(list(unfold(items2)))  
print(list(unfold(items3)))
```

<sup>②</sup> 思路来源 [http://python3-cookbook.readthedocs.io/zh\\_CN/latest/c04/p14\\_flattening\\_nested\\_sequence.html](http://python3-cookbook.readthedocs.io/zh_CN/latest/c04/p14_flattening_nested_sequence.html)

## 第三章 算法

### 问题 1 简单算法