argparse 模块

一个可执行文件或者脚本都可以接收参数。

```
$ ls -l /etc
/etc 是位置参数
-l 是短选项
```

如何把这些参数传递给程序呢？
从3.2开始Python提供了参数分析的模块argparse。

参数分类

参数分为：

- 位置参数，参数放在那里，就要对应一个参数位置。例如/etc就是对应一个参数位置。
- 选项参数，必须通过前面是`-`的短选项或者`--`的长选项，然后后面的才算该选项的参数，当然选项后面也可以没有参数。

上例中，/etc对应的是位置参数，-l是选项参数。

`ls -alh src`

基本解析

先来一段最简单的程序

```
import argparse

parser = argparse.ArgumentParser() # 获得一个参数解析器
args = parser.parse_args()  # 分析参数
parser.print_help()  # 打印帮助
```

运行结果

```
$ python test.py -h
usage: test1.py [-h]

optional arguments:
  -h, --help  show this help message and exit
```

argparse不仅仅做了参数的定义和解析，还自动帮助生成了帮助信息。尤其是usage，可以看到现在定义的参数是否是自己想要的。

解析器的参数

参数名称	说明
prog	程序的名字，缺省使用 sys.argv[0] 的 basename
add_help	自动为解析器增加`-h`和`--help`选项，默认为True
description	为程序功能添加描述

`parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory contents')`

```
$ python test.py --help
usage: ls [-h]

list directory contents

optional arguments:
  -h, --help  show this help message and exit
```

位置参数解析

ls 基本功能应该解决目录内容的打印。
打印的时候应该指定目录路径，需要位置参数。

```
import argparse

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory
contents')
parser.add_argument('path')

args = parser.parse_args()  # 分析参数
parser.print_help()  # 打印帮助

# 运行结果，出现了错误，提示需要输入path对应的位置参数
usage: ls [-h] path
ls: error: the following arguments are required: path
```

程序定义为：
ls [-h] path
-h为帮助选项，可有可无
path为位置参数，必须提供

传参

parse_args(args=None, namespace=None)
args 参数列表，一个可迭代对象。内部会把可迭代对象转换成list。如果为None则使用命令行传入参数，非None则使用args参数的可迭代对象。

```
import argparse
```

```
# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory
contents')
parser.add_argument('path') # 位置参数

args = parser.parse_args(('/etc',))  # 分析参数，同时传入可迭代的参数
print(args, args.path) # 打印名词空间中收集的参数
parser.print_help()  # 打印帮助

运行结果
Namespace(path='/etc') /etc
usage: ls [-h] path

list directory contents

positional arguments:
  path

optional arguments:
  -h, --help  show this help message and exit
```

Namespace(path='/etc')里面的path参数存储在了一个Namespace对象内的属性上，可以通过Namespace对象属性来访问，例如args.path

非必须位置参数

上面的代码必须输入位置参数，否则会报错。

```
usage: ls [-h] path
ls: error: the following arguments are required: path
```

但有时候，ls命令不输入任何路径的话就表示列出当前目录的文件列表。

```
import argparse

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory
contents')
parser.add_argument('path', nargs='?', default='.', help="path help") # 位置参数，可有可无，缺省值，帮助

args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
print(args) # 打印名词空间中收集的参数
parser.print_help()  # 打印帮助

# 运行结果
Namespace(path='.')
usage: ls [-h] [path]

list directory contents
```

```
positional arguments:
  path        path help

optional arguments:
  -h, --help  show this help message and exit
```

可以看出path也变成**可选**的位置参数，没有提供就使用默认值`.点号`表示当前路径。

- help 表示帮助文档中这个参数的描述
- nargs 表示这个参数接收结果参数
  - ? 表示可有可无
  - + 表示至少一个
  - * 可以任意个
  - 数字表示必须是指定数目个
- default 表示如果不提供该参数，就使用这个值。一般和？、*配合，因为它们都可以不提供位置参数，不提供就是用缺省值

选项参数

-l的实现

`parser.add_argument('-l')`就增加了选项参数，参数定义为
`ls [-h][-l L] [path]`
和我们要的形式有一点出入，我们期望的是[-h]，怎么解决？

nargs能够解决吗？
parser.add_argument('-l', nargs='?')
`ls [-h][-l [L]] [path]`
L变成了可选，而不是-l。

那么，直接把nargs=0，意思就是让这个选项接收0个参数，如下
`parser.add_argument('-l', nargs=0)`
结果，抛出异常

> raise ValueError('nargs for store actions must be > 0; if you '
> ValueError: nargs for store actions must be > 0; if you have nothing to store, actions such as store true or store const may be more appropriate

看来nargs是控制位置参数和选项参数的。

为了这个问题，使用**action参数**
`parser.add_argument('-l', action='store_true')`
看到命令定义变成了`ls [-h] [-l] [path]`
提供-l选项，例如
ls -l得到Namespace(l=True, path='.')，提供-l值是True
ls 得到Namespace(l=False, path='.')，提供-l值是False
这样同True、False来判断用户是否提供了该选项

`parser.add_argument('-l', action='store_const', const = 20)`表示，如果提供-l选项，则对应的值是20，如果不提供，对应值是None

-a的实现

parser.add_argument('-a', '--all', action='store_true')
长短选项可以同时给出。

代码

```
import argparse

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory
contents')
parser.add_argument('path', nargs='?', default='.', help="directory") # 位置参数，可有可无，缺省
值，帮助
parser.add_argument('-l', action='store_true', help='use a long listing format')
parser.add_argument('-a', '--all', action='store_true', help='show all files, do not ignore
entries starting with .')

args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
print(args) # 打印名词空间中收集的参数
parser.print_help()  # 打印帮助

# 运行结果
Namespace(all=False, l=False, path='.')
usage: ls [-h] [-l] [-a] [path]

list directory contents

positional arguments:
  path        directory

optional arguments:
  -h, --help  show this help message and exit
  -l          use a long listing format
  -a, --all   show all files, do not ignore entries starting with .

# parser.parse_args('-l -a /tmp'.split())运行结果
Namespace(all=True, l=True, path='/tmp')
```

---

ls业务功能的实现

到目前为止，已经解决了参数的定义和传参的问题，下面就要解决业务问题：

1. 列出所有指定路径的文件，默认是不递归的
2. -a 显示所有文件，包括隐藏文件
3. -l 详细列表模式显示

代码实现

```
import argparse
```

```python
from pathlib import Path
from datetime import datetime

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory contents')
parser.add_argument('path', nargs='?', default='.', help="directory") # 位置参数，可有可无，缺省值，帮助
parser.add_argument('-l', action='store_true', help='use a long listing format')
parser.add_argument('-a', '--all', action='store_true', help='show all files, do not ignore entries starting with .')

args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
print(args) # 打印名词空间中收集的参数
parser.print_help()  # 打印帮助


def listdir(path, all=False):
    """列出本目录文件"""
    p = Path(path)
    # for f in p.iterdir():
    #     if not all and f.name.startswith('.'): # 不显示隐藏文件
    #         continue
    #     yield f.name
    #yield from filter(lambda f:not(not all and f.name.startswith('.')), p.iterdir())
    #yield from filter(lambda f: all or not f.name.startswith('.'), p.iterdir())
    yield from map(str, filter(lambda f:all or not f.name.startswith('.'), p.iterdir()))

print(list(listdir(args.path)))

# 获取文件类型
def _getfiletype(f:Path):
    if f.is_dir():
        return 'd'
    elif f.is_block_device():
        return 'b'
    elif f.is_char_device():
        return 'c'
    elif f.is_socket():
        return 's'
    elif f.is_symlink():
        return 'l'
    else:
        return '-'

# -rw-rw-r-- 1 python python     5 Oct 25 00:07 test4
def listdirdetail(path, all=False):
    """详细列出本目录"""
    p = Path(path)
    for f in p.iterdir():
        if not all and f.name.startswith('.'): # 不显示隐藏文件
            continue
        # mode 硬链接 属主 属组 字节 时间 name
```

```
        stat = f.stat()
        t = _getfiletype(f)
        mode = oct(stat.st_mode)[-3:]
        mtime = datetime.fromtimestamp(stat.st_mtime).strftime('%Y %m %d %H:%M:%S')
        yield (t, mode, stat.st_nlink, stat.st_uid, stat.st_gid, stat.st_size, mtime, f.name)


print(list(listdirdetail(args.path)))
```

mode是整数，八进制描述的权限，最终显示为rwx的格式。

方法1

```
modelist = list('rwx' * 3)
def _getmodestr(mode:int):
    m = mode & 0o777
    # print(m)
    # print(m, bin(m), oct(m))
    mstr = ""
    for i, v in enumerate(bin(m)[-9:]):
        mstr += modelist[i] if v == '1' else '-'
    return mstr
```

方法2 移位

```
modelist = list('rwx' * 3)
def _getmodestr(mode:int):
    m = mode & 0o777
    mstr = ""
    for i in range(8, -1, -1):
        mstr += modelist[8-i] if m >> i & 1 else '-'
    return mstr
```

合并列出文件函数

listdirdetail和listdir几乎一样，重复太多，合并

```
import argparse
from pathlib import Path
from datetime import datetime

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory
contents')
parser.add_argument('path', nargs='?', default='.', help="directory") # 位置参数，可有可无，缺省
值，帮助
parser.add_argument('-l', action='store_true', help='use a long listing format')
parser.add_argument('-a', '--all', action='store_true', help='show all files, do not ignore
entries starting with .')
```

```
args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
print(args) # 打印名词空间中收集的参数
parser.print_help()  # 打印帮助


def _getfiletype(f:Path):
    """获取文件类型"""
    if f.is_dir():
        return 'd'
    elif f.is_block_device():
        return 'b'
    elif f.is_char_device():
        return 'c'
    elif f.is_socket():
        return 's'
    elif f.is_symlink():
        return 'l'
    elif f.is_fifo(): # pipe
        return 'p'
    else:
        return '-'


modelist = dict(zip(range(9),['r','w','x','r','w','x','r','w','x']))
def _getmodestr(mode:int):
    m = mode & 0o777
    mstr = ''
    for i in range(8,-1,-1):
        if m >> i & 1:
            mstr += modelist[8-i]
        else:
            mstr += '-'
    return mstr


def listdir(path, all=False, detail=False):
    """详细列出本目录"""
    p = Path(path)
    for i in p.iterdir():
        if not all and i.name.startswith('.'): # 不显示隐藏文件
            continue
        if not detail:
            yield (i.name,)
        else:
            # -rw-rw-r-- 1     python python     5 Oct 25 00:07 test4
            # mode       硬链接 属主   属组       字节 时间         name
            stat = i.stat()
            mode = _getfiletype(i) + _getmodestr(stat.st_mode)
            atime = datetime.fromtimestamp(stat.st_atime).strftime('%Y %m %d %H:%M:%S')
            yield (mode, stat.st_nlink, stat.st_uid, stat.st_gid, stat.st_size, atime, i.name)


for x in listdir(args.path):
```

```
    print(x)
```

排序

ls的显示是把文件名按照升序排序输出。

```
# 排序
sorted(listdir(args.path, detail=True), key=lambda x: x[len(x) - 1])
```

完整代码

再次重构代码

```
import argparse
from pathlib import Path
from datetime import datetime

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=True, description='list directory
contents')
parser.add_argument('path', nargs='?', default='.', help="directory") # 位置参数，可有可无，缺省
值，帮助
parser.add_argument('-l', action='store_true', help='use a long listing format')
parser.add_argument('-a', '--all', action='store_true', help='show all files, do not ignore
entries starting with .')

def listdir(path, all=False, detail=False):
    # 获取文件类型
    def _getfiletype(f:Path):
        if f.is_dir():
            return 'd'
        elif f.is_block_device():
            return 'b'
        elif f.is_char_device():
            return 'c'
        elif f.is_socket():
            return 's'
        elif f.is_symlink():
            return 'l'
        else:
            return '-'

    modelist = list('rwx' * 3)

    def _getmodestr(mode:int):
        m = mode & 0o777
        mstr = ""
        for i in range(8, -1, -1):
            mstr += modelist[8-i] if m >> i & 1 else '-'
        return mstr
```

```
    def _listdir(path, all=False, detail=False):
        """详细列出本目录"""
        p = Path(path)
        for f in p.iterdir():
            if not all and f.name.startswith('.'): # 不显示隐藏文件
                continue
            if not detail:
                yield (f.name,)
            else:
                # -rw-rw-r-- 1 python python     5 Oct 25 00:07 test4
                # mode 硬链接 属主 属组 字节 时间 name
                stat = f.stat()
                t = _getfiletype(f)
                mode = _getmodestr(stat.st_mode)
                mtime = datetime.fromtimestamp(stat.st_mtime).strftime('%Y %m %d %H:%M:%S')
                yield (t, mode, stat.st_nlink, stat.st_uid, stat.st_gid, stat.st_size, mtime,
f.name)

    # 排序
    yield from sorted(_listdir(path, all, detail), key=lambda x:x[len(x)-1])

if __name__ == '__main__':
    args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
    print(args)  # 打印名词空间中收集的参数
    parser.print_help()  # 打印帮助
    files = listdir(args.path, args.all, args.l)
    print(list(files))
```

-h的实现

-h, --human-readable，如果-l存在，-h有效。

1、 增加选项参数

```
parser = argparse.ArgumentParser(prog='ls', description='list directory contents',
add_help=False)
parser.add_argument('-h', '--human-readable', action='store_true', help='with -l, print sizes in
human readable format')
```

2、增加一个函数，能够解决单位转换的

```
def _gethuman(size: int):
    units = ' KMGTP'
    depth = 0
    while size > 1000 and depth + 1 < len(units):
        # 当前size大于1000，且depth不是最后一个
        size = size // 1000
        depth += 1

    return "{}{}".format(size, units[depth])
```

3、在-l逻辑部分增加处理

```
size = stat.st_size if not human else _gethuman(stat.st_size)
```

其他的完善

uid、gid的转换

pwd模块，The password database，提供访问Linux、Unix 的 password文件的方式。windows没有。
pwd.getpwuid(Path().stat().st_uid).pw_name
grp模块，Linux、Unix获取组信息的模块。windows没有
grp.getgrgid(Path().stat().st_gid).gr_name

pathlib模块，Path().group() 或者 Path().owner()也可以，本质上它们就是调用pwd模块和grp模块。

由于windows不支持，这次可以不加这个uid、gid的转换

代码改进

```
import argparse
from pathlib import Path
from datetime import datetime

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=False, description='list directory
contents')
parser.add_argument('path', nargs='?', default='.', help="directory") # 位置参数，可有可无，缺省
值，帮助
parser.add_argument('-l', action='store_true', help='use a long listing format')
parser.add_argument('-a', '--all', action='store_true', help='show all files, do not ignore
entries starting with .')
parser.add_argument('-h', '--human-readable', action='store_true', help='with -l, print sizes in
human readable format')

def listdir(path, all=False, detail=False, human=False):
    # 获取文件类型
    def _getfiletype(f:Path):
        if f.is_dir():
            return 'd'
        elif f.is_block_device():
            return 'b'
        elif f.is_char_device():
            return 'c'
        elif f.is_socket():
            return 's'
        elif f.is_symlink():
            return 'l'
        else:
            return '-'

    modelist = list('rwx' * 3)
```

```
    def _getmodestr(mode:int):
        m = mode & 0o777
        mstr = ""
        for i in range(8, -1, -1):
            mstr += modelist[8-i] if m >> i & 1 else '-'
        return mstr

    def _gethuman(size: int):
        units = ' KMGTP'
        depth = 0
        while size > 1000 and depth + 1 < len(units):
            # 当前size大于1000，且depth不是最后一个
            size = size // 1000
            depth += 1

        return "{}{}".format(size, units[depth])

    def _listdir(path, all=False, detail=False, human=False):
        """详细列出本目录"""
        p = Path(path)
        for f in p.iterdir():
            if not all and f.name.startswith('.'): # 不显示隐藏文件
                continue
            if not detail:
                yield (f.name,)
            else:
                # -rw-rw-r-- 1 python python     5 Oct 25 00:07 test4
                # mode 硬链接 属主 属组 字节 时间 name
                stat = f.stat()
                t = _getfiletype(f)
                mode = _getmodestr(stat.st_mode)
                mtime = datetime.fromtimestamp(stat.st_mtime).strftime('%Y %m %d %H:%M:%S')
                size = stat.st_size if not human else _gethuman(stat.st_size)
                yield (t, mode, stat.st_nlink, stat.st_uid, stat.st_gid, size, mtime, f.name)

    # 排序
    yield from sorted(_listdir(path, all, detail), key=lambda x:x[-1])

if __name__ == '__main__':
    args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
    print(args)  # 打印名词空间中收集的参数
    parser.print_help()  # 打印帮助
    files = listdir(args.path, args.all, args.l)
    print(list(files))
```

改进mode的方法

使用stat模块

```
import stat
from pathlib import Path
stat.filemode(Path().stat().st_mode)
```

最终代码

```
import argparse
from pathlib import Path
from datetime import datetime
import stat

# 获得一个参数解析器
parser = argparse.ArgumentParser(prog='ls', add_help=False, description='list directory
contents')
parser.add_argument('path', nargs='?', default='.', help="directory") # 位置参数，可有可无，缺省
值，帮助
parser.add_argument('-l', action='store_true', help='use a long listing format')
parser.add_argument('-a', '--all', action='store_true', help='show all files, do not ignore
entries starting with .')
parser.add_argument('-h', '--human-readable', action='store_true', help='with -l, print sizes in
human readable format')

def listdir(path, all=False, detail=False, human=False):
    def _gethuman(size: int):
        units = ' KMGTP'
        depth = 0
        while size > 1000 and depth + 1 < len(units):
            # 当前size大于1000，且depth不是最后一个
            size = size // 1000
            depth += 1

        return "{}{}".format(size, units[depth])

    def _listdir(path, all=False, detail=False, human=False):
        """详细列出本目录"""
        p = Path(path)
        for f in p.iterdir():
            if not all and f.name.startswith('.'): # 不显示隐藏文件
                continue
            if not detail:
                yield (f.name,)
            else:
                # -rw-rw-r-- 1     python python     5 Oct 25 00:07  test4
                # mode       硬链接 属主   属组       字节 时间         name
                st = f.stat()
                # t = _getfiletype(f)
                # mode = _getmodestr(stat.st_mode)
                mode = stat.filemode(st.st_mode)
                mtime = datetime.fromtimestamp(st.st_mtime).strftime('%Y %m %d %H:%M:%S')
                size = st.st_size if not human else _gethuman(st.st_size)
                yield (mode, st.st_nlink, st.st_uid, st.st_gid, size, mtime, f.name)

    # 排序
    yield from sorted(_listdir(path, all, detail, human), key=lambda x:x[-1])

if __name__ == '__main__':
    args = parser.parse_args()  # 分析参数，同时传入可迭代的参数
```

```
    print(args)  # 打印名词空间中收集的参数
    parser.print_help()  # 打印帮助
    files = listdir(args.path, args.all, args.l, args.human_readable)
    print(list(files))
```

测试

```
$ python xxx.py -lah
$ python xxx.py /etc/ -lah
```

思考
如果要实现ls -lah /etc /tmp /home 怎么实现