user

Python之OS模块39个常用函数详解

assistant

Python 的 `os` 模块提供了与操作系统交互的功能，包括文件和目录操作、环境变量管理、进程控制等。以下是 39 个常用函数的详细说明（基于 Python 3.x，特别是兼容 Python 3.10+ 的常见用法）：

一、路径操作（Path Operations）

1. `os.getcwd()`
   获取当前工作目录（Current Working Directory）。

```
print(os.getcwd())  # e.g., '/home/user'
```

2. `os.chdir(path)`
   改变当前工作目录。

```
os.chdir('/tmp')
```

3. `os.path.abspath(path)`
   返回 path 的绝对路径。

```
os.path.abspath('test.txt')  # '/home/user/test.txt'
```

4. `os.path.join(path, *paths)`
   智能拼接路径（跨平台兼容）。

```
os.path.join('folder', 'sub', 'file.txt')  # 'folder/sub/file.txt' (Linux)
```

5. `os.path.basename(path)`
   返回路径中的文件名部分。

```
os.path.basename('/home/user/file.txt')  # 'file.txt'
```

6. `os.path.dirname(path)`
   返回路径中的目录部分。

```
os.path.dirname('/home/user/file.txt')  # '/home/user'
```

7. os.path.split(path)
   将路径拆分为 (目录, 文件名) 元组。

```
os.path.split('/a/b/c.txt')  # ('/a/b', 'c.txt')
```

8. os.path.splitext(path)
   分离文件名和扩展名。

```
os.path.splitext('image.png')  # ('image', '.png')
```

9. os.path.exists(path)
   判断路径是否存在。

```
os.path.exists('data.txt')  # True/False
```

10. os.path.isfile(path)
    判断是否为文件。

11. os.path.isdir(path)
    判断是否为目录。

12. os.path.islink(path)
    判断是否为符号链接（软链接）。

13. os.path.getsize(path)
    获取文件大小（字节）。

14. os.path.getmtime(path)
    获取文件最后修改时间（时间戳）。

15. os.path.normpath(path)
    规范化路径（如处理 `..` 和 `.`）。

二、目录与文件操作

16. os.listdir(path='.')
    列出指定目录下的文件和子目录名（不递归）。

```
os.listdir('.')  # ['file1.txt', 'dir1']
```

17. os.mkdir(path, mode=0o777)
    创建单层目录（若父目录不存在则报错）。

18. os.makedirs(path, mode=0o777, exist_ok=False)
    递归创建多层目录（类似 `mkdir -p`）。

```
os.makedirs('a/b/c', exist_ok=True)
```

19. os.rmdir(path)
    删除空目录。

20. os.removedirs(path)
    递归删除空目录（从最内层开始删，遇到非空停止）。

21. os.remove(path) 或 os.unlink(path)
    删除文件（两者功能相同， unlink 是 Unix 风格名称）。

22. os.rename(src, dst)
    重命名或移动文件/目录。

23. os.replace(src, dst)
    类似 rename ，但在 Windows 上可覆盖目标（原子操作）。

24. os.walk(top, topdown=True, ...)
    递归遍历目录树，生成 (dirpath, dirnames, filenames) 。

```
for root, dirs, files in os.walk('/home'):
    for f in files:
        print(os.path.join(root, f))
```

三、权限与属性

25. os.chmod(path, mode)
    修改文件或目录权限（如 0o755 ）。

```
os.chmod('script.sh', 0o755)
```

26. os.stat(path)
    获取文件状态信息（返回 os.stat_result 对象）。

```
st = os.stat('file.txt')
print(st.st_size, st.st_mtime)
```

27. os.access(path, mode)
    检查文件访问权限（ os.R_OK , os.W_OK , os.X_OK ）。

```
if os.access('config.ini', os.R_OK):
    print("Readable")
```

四、环境变量与系统信息

28. os.environ
    环境变量字典（可读写）。

```
print(os.environ['HOME'])
os.environ['MY_VAR'] = 'value'
```

29. os.getenv(key, default=None)
    安全获取环境变量（推荐方式）。

```
home = os.getenv('HOME', '/default')
```

30. os.putenv(key, value)
    设置环境变量（但修改 os.environ 更可靠）。

31. os.name
    返回操作系统标识：'posix'（Linux/macOS）、'nt'（Windows）。

32. ** os.uname() **（仅 POSIX）
    返回系统信息（系统名、主机名、内核版本等）。

33. os.cpu_count()
    返回 CPU 核心数。

五、进程与执行

34. os.system(command)
    执行系统命令（简单但不安全，建议用 subprocess ）。

```
os.system('ls -l')
```

35. os.exec*() 系列（如 os.execl，os.execv ）
    替换当前进程（Unix 常用，Windows 支持有限）。

36. ** os.fork() **（仅 Unix）
    创建子进程（返回 0 给子进程，子 PID 给父进程）。

37. os.getpid()
    获取当前进程 ID。

38. os.getppid()
    获取父进程 ID。

39. os.kill(pid, sig)
    向进程发送信号（如 signal.SIGTERM ）。

补充说明

- 路径操作推荐使用 os.path 或更现代的 pathlib.Path （Python 3.4+）。
- 文件操作建议结合 open()、 shutil （高级文件操作）和 subprocess （执行命令）。
- 在 Windows 上某些函数（如 fork，uname ）不可用。

✅ 最佳实践建议：

- 使用 os.path.join 而非字符串拼接路径。
- 优先使用 pathlib （面向对象、更简洁）：

```
from pathlib import Path
p = Path('data') / 'file.txt'
```

- 敏感操作（如删除、重命名）应先检查 os.path.exists()。

如需示例代码或某函数深入解析，可继续提问！