struct模块

07-文件 page32-34

struct模块(9.3.2)

拓展的内容

- 在网络编程、与采用C语言编写的应用程序交互时进行序列化和反序列化
- values--> bytes：pack(fmt, v1, v2, ...)：将v1、v2等值按照fmt给出的格式进行转换，返回一个bytes对象
- bytes --> values: unpack(fmt, buffer)：按照fmt给出的格式将bytes对象buffer中的内容进行转换，返回包含了多个值的**元组**
- 字节顺序、大小和对齐方式：基于fmt的第1个字符的取值：
  - 网络字节或大端：高字节在先的顺序，0x1122--> 11 22
  - 小端：高字节在后的顺序，0x1122 --> 2211
  - 对齐：比如整数对应的地址为4的倍数的边界

字符	字节顺序	大小和对齐
@或非@=<>!字符	native	native
=	native	标准大小，无对齐
!或>	大端(big-endian)	标准大小，无对齐
<	小端(little-endian)	标准大小，无对齐

数值类型：小写/大写，对应有符号和无符号类型)

格式	C类型	Python	标准字节数
x	填充字节	无对应	
c	char	长度1的bytes	1
?	_Bool (C99)	bool	1
bB	signed/unsigned char	integer	1
hH	[unsigned] short	integer	2
iI	[unsigned] int	integer	4
lL	[unsigned] long	integer	4
qQ	[unsigned] long long	integer	8
nN	ssize_t, size_t	integer	native
f	float	float	4
d	double	float	8
p	char[]	bytes	
s	char[]	bytes	
P	void *	integer	

- 3c相当于ccc，数字3表示重复次数。注意3s和sp

```
>>> struct.pack('10s', b'abc')
b'abc\x00\x00\x00\x00\x00\x00\x00'
>>> struct.pack('10p', b'def')  # 第一个字节为长度
b'\x03def\x00\x00\x00\x00\x00\x00'
```

struct模块

- calcsize(fmt)：根据fmt，计算如果要pack需要的字节数
- 对齐：有些系统要求：
  - int对应的地址为4的倍数的边界
  - short对应的地址为2的倍数的边界
- 大端和小端：
  - 网络字节或大端：高字节在先的顺序，0x1122--> 11 22
  - 小端：高字节在后的顺序，0x1122 --> 2211

字符	字节顺序	大小和对齐
@或非@=<>!字符	native	native
=	native	标准大小，无对齐
!或>	大端(big-endian)	标准大小，无对齐
<	小端(little-endian)	标准大小，无对齐

拓展的内容

```
>>> import sys
>>> sys.byteorder
'little'
>>> struct.pack('3i', 1, 2, 3)                    native
b'\x01\x00\x00\x00\x02\x00\x00\x00\x03\x00\x00\x00'
>>> struct.pack('@3i', 1, 2, 3)                   native
b'\x01\x00\x00\x00\x02\x00\x00\x00\x03\x00\x00\x00'
>>> struct.pack('@bhi', 1, 2, 3)      Native，可看到填充
b'\x01\x00\x02\x00\x03\x00\x00\x00'

>>> struct.pack('<bhi', 1, 2, 3)  # 采用小端字节顺序   小端
b'\x01\x02\x00\x03\x00\x00\x00'
>>> struct.pack('>bhi', 1, 2, 3)  # 采用大端或网络字节顺序
b'\x01\x00\x02\x00\x00\x00\x03'                    大端
>>> struct.pack('!bhi', 1, 2, 3)
b'\x01\x00\x02\x00\x00\x00\x03'

>>> struct.calcsize('bhi')                        native
8
# why? 在实验主机的native实现时，short要以地址为2的
# 倍数的边界开始。int要以地址为4的倍数的边界开始
>>> struct.pack('<bxhi', 1, 2, 3)
b'\x01\x00\x02\x00\x03\x00\x00\x00'
```

struct模块

拓展的内容

- pack_into(fmt, buffer, offset, v1, v2, ...): 类似于pack，只是不是返回bytes对象，而是写入到bytearray对象buffer中，从offset开始写入
- unpack_from(fmt, buffer, offset=0)：类似于unpack，只是从buffer的偏移量offset处开始unpack
- 字符串：
  - 首先转换为字节串，计算字节串的长度，假设为s。
  - f'{s}s': 字节串
  - f'{s+1}s': 字节串 + NULL，与C语言的NULL字符结尾的字符串char[]对应
  - f'{s+1}p': 长度+字节串

struct_demo.py

```
i, f, b, s = 1234567, 3.14, True, 'python程序设计'
s2 = s.encode()
s2_len = len(s2)
packed_buffer = struct.pack(f'if?h{s2_len}s', i, f, b, s2_len, s2)
print(packed_buffer)
```

```
offset = struct.calcsize('if?h')
values = struct.unpack('if?h', packed_buffer[:offset])
print(*values)
s = struct.unpack_from(f'{values[-1]}s', packed_buffer, offset)
print(s[0].decode())
```