atoi 和 itoa 函数实现

atoi 函数是 C 语言库提供的，是把字符串转换成整型数和把字符串转换成整型数。而 itoa 函数是广泛应用的非标准 C 语言扩展函数,由于它不是标准 C 语言函数，所以不能在所有的编译器中使用，它的功能是把一整数转换为字符串。
两个函数功能很好理解，但是它们的实现需要考虑到很多问题，在面试中，很多面试官都会问 atoi 和 itoa 的实现，这可以很好的了解程序员编程的功底。
那么在实现 atoi 一般需要考虑到那些情况呢？

1. 首先是整数，很多人很自然的会忽略负整数。
2. 其次是考虑到上溢和下溢
3. 字符串以空格开始怎么去处理
4. 字符串中包含了除 0-9 之间的字符该怎么去处理

实现

```
01 #include <stdio.h>
02
03 #define MAX_INT ((1 << 31) - 1)
04 #define MIN_INT (-(1 << 31))
05
06 int atoi(const char *str){
07     char *temp = str;
08     int i = 0;
09     int flags = 0;
10     unsigned int sum = 0;
11     while(*temp == ' ') ++temp ;
12     if(*temp != '-' && *temp != '+' && (*temp < '0' || *temp > '9')){//第一个字符不是数字
13         return 0;
14     }
15
16     if(*temp == '-'){ //第一个是负号
17         flags = 1;
18         ++temp;
```

```
19    }else if(*temp == '+'){
20        ++temp;
21    }
22
23    while(*temp >= '0' && *temp <= '9'){
24        if(!flags){//上溢
25            if(sum > MAX_INT / 10 || (sum == MAX_INT / 10 && (*temp > '7'))){
26                return MAX_INT;
27            }
28
29        }else{//下溢
30            if(sum > MAX_INT / 10 || (sum == MAX_INT / 10 && (*temp > '8'))){
31                return MIN_INT;
32            }
33        }
34
35        sum = sum * 10 + (*temp - '0');
36        ++temp;
37    }
38
39    //if(flags){
40        //sum *= -1;
41    //}
42    return flags ? (-1 * sum) : sum;
43 }
44
45 int main(){
46    printf("%d\n", atoi("     0125464 c 646"));
47    return 0;
48 }
```

ansi 库中的实现：

```
001 #include    <ctype.h>
002 #include    <stdlib.h>
```

```
003
004 int
005 atoi(register const char *nptr)
006 {
007     return strtol(nptr, (char **) NULL, 10);
008 }
009
010 /*
011  * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
012  * See the copyright notice in the ACK home directory, in the file "Copyright".
013  */
014 /* $Header: strtol.c,v 1.4 90/05/11 15:22:19 eck Exp $ */
015
016 #include     <ctype.h>
017 #include     <errno.h>
018 #include     <limits.h>
019 #include     <stdlib.h>
020
021 static unsigned long
022 string2long(register const char *nptr, char **endptr,
023             int base, int is_signed);
024
025 long int
026 strtol(register const char *nptr, char **endptr, int base)
027 {
028     return (signed long)string2long(nptr, endptr, base, 1);
029 }
030
031 #define between(a, c, z)    ((unsigned) ((c) - (a)) <= (unsigned) ((z) - (a)))
032
033 static unsigned long
034 string2long(register const char *nptr, char ** const endptr,
035             int base, int is_signed)
036 {
```

```
037  register unsigned int v;
038  register unsigned long val = 0;
039  register int c;
040  int ovfl = 0, sign = 1;
041  const char *startnptr = nptr, *nrstart;
042
043  if (endptr) *endptr = (char *)nptr;
044  while (isspace(*nptr)) nptr++;
045  c = *nptr;
046
047  if (c == '-' || c == '+') {
048     if (c == '-') sign = -1;
049     nptr++;
050  }
051  nrstart = nptr;        /* start of the number */
052
053  /* When base is 0, the syntax determines the actual base */
054  if (base == 0)
055     if (*nptr == '0')
056        if (*++nptr == 'x' || *nptr == 'X') {
057           base = 16;
058           nptr++;
059        }
060        else     base = 8;
061     else     base = 10;
062  else if (base==16 && *nptr=='0' && (*++nptr =='x' || *nptr =='X'))
063     nptr++;
064
065  for (;;) {
066     c = *nptr;
067     if (between('0', c, '9')) {
068        v = c - '0';
069     } else
070     if (between('a', c, 'z')) {
```

```
071         v = c - 'a' + 0xa;
072       } else
073       if (between('A', c, 'Z')) {
074         v = c - 'A' + 0xA;
075       } else {
076         break;
077       }
078       if (v >= base) break;
079       if (val > (ULONG_MAX - v) / base) ovfl++;
080       val = (val * base) + v;
081       nptr++;
082     }
083     if (endptr) {
084       if (nrstart == nptr) *endptr = (char *)startnptr;
085       else *endptr = (char *)nptr;
086     }
087
088     if (!ovfl) {
089       /* Overflow is only possible when converting a signed long. */
090       if (is_signed
091         && (    (sign < 0 && val > -(unsigned long)LONG_MIN)
092         || (sign > 0 && val > LONG_MAX)))
093         ovfl++;
094     }
095
096     if (ovfl) {
097       errno = ERANGE;
098       if (is_signed)
099         if (sign < 0) return LONG_MIN;
100         else return LONG_MAX;
101       else return ULONG_MAX;
102     }
103     return (long) sign * val;
104 }
```

itoa 实现注意事项：

1. 忘记考虑负数；
2. 忘记在末尾加上'\0'。

实现：

```
01 #include <stdlib.h>
02 #include <stdio.h>
03
04 char *itoa(int num,char *str,int radix){//num：int 型原数,str:需转换成的 string，radix,原进制，
05
06   /* 索引表 */
07   char index[]="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
08   unsigned unum;/* 中间变量 */
09   int i=0,j,k;
10
11   /* 确定 unum 的值 */
12   if(radix==10&&num<0){/* 十进制负数 */
13     unum=(unsigned)-num;
14     str[i++]='-';
15   }else
16     unum=(unsigned)num;/* 其他情况 */
17   /* 逆序 */
18   do{
19     str[i++]=index[unum%(unsigned)radix];
20     unum/=radix;
21   }while(unum);
22
23   str[i]='\0';
24   /* 转换 */
25   if(str[0]=='-')
26     k=1;/* 十进制负数 */
27   else
```

```
28      k=0;
29    /* 将原来的“/2”改为“/2.0”，保证当 num 在 16~255 之间，radix 等于 16 时，也能得到正确结果 */
30    for(j=k;j<(i-1)/2.0+k;j++){
31      num=str[j];
32      str[j]=str[i-j-1+k];
33      str[i-j-1+k]=num;
34    }
35    return str;
36 }
37
38 int main(){
39
40    int number = 0177777777777777777774;
41    char string[25];
42
43    itoa(number, string, 10);
44
45    printf("integer = %d string = %s\n", number, string);
46    return 0;
47 }
```

ansi 库实现

```
01 #define NUMBER_OF_DIGITS 16
02
03 void _uitoa(unsigned int value, char* string, unsigned char radix)
04 {
05 unsigned char index, i;
06   /* char buffer[NUMBER_OF_DIGITS]; */ /* space for NUMBER_OF_DIGITS + '\0' */
07
08   index = NUMBER_OF_DIGITS;
09   i = 0;
10
11   do {
```

```
12    string[--index] = '0' + (value % radix);
13    if ( string[index] > '9') string[index] += 'A' - '9' - 1;
14    value /= radix;
15   } while (value != 0);
16
17   do {
18    string[i++] = string[index++];
19   } while ( index < NUMBER_OF_DIGITS );
20
21   string[i] = 0; /* string terminator */
22 }
23
24 void _itoa(int value, char* string, unsigned char radix)
25 {
26   if (value < 0 && radix == 10) {
27    *string++ = '-';
28    _uitoa(-value, string, radix);
29   }
30   else {
31    _uitoa(value, string, radix);
32   }
33 }
```

云帆教育大数据学院 www.cloudyhadoop.com

通过最新实战课程，系统学习 hadoop2.x 开发技能，在云帆教育，课程源于企业真实需求，最有实战价值，成为正式会员，可无限制在线学习全部教程；培训市场这么乱，云帆大数据值得你选择!!详情请加入 QQ 群：**374152400**，咨询课程顾问!

云帆教育大数据学院

关注云帆教育微信公众号 yfteach，第一时间获取公开课信息。