首先感谢如下两篇的blog，让我走出了很大的一个误区：

http://www.cppblog.com/kongque/archive/2011/01/18/138765.aspx
http://blog.csdn.NET/zjwoody/article/details/7882240

在我的一个项目中，因为需要与串口通信，每次读写都需要延时usleep(1000)=1ms，但是通信量非常大，每一次工作这样的通信大概有300次左右，这样算下耗时应该是300ms左右。

但是通过strace打印出系统函数调用发现实际接近900ms，仔细观察strace日志才发现，每次usleep(1000000)其实都延时了2ms，之后上网搜索才发现usleep是不精确的。

[cpp] view plain copy
print?

```
1. 1.sleep的精度是秒
2. 2.usleep的精度是微妙，不精确
3. 3.select的精度是微妙，精确
4.     struct timeval delay;
5.     delay.tv_sec = 0;
6.     delay.tv_usec = 20 * 1000; // 20 ms
7.     select(0, NULL, NULL, NULL, &delay);
8.
9. 4.nanosleep的精度是纳秒，不精确
10.
11. unix、linux系统尽量不要使用usleep和sleep而应该使用nanosleep，使用nanosleep应注意判断返回值和错误代码，否则容易造成cpu占用率100%。
```

这是第一篇blog中提到的，然后第二篇blog中提供的测试代码，本人做了少量改动(原作者没有打印出usleep(0)时的信息)，代码如下：

[cpp] view plain copy
print?

```
1. /*
```

```
2.          make:  gcc -o test_sleep test_sleep.c
3.  */
4.  #include <stdio.h>
5.  #include <stdlib.h>
6.  #include <time.h>
7.  #include <sys/time.h>
8.  #include <errno.h>
9.  #include <string.h>
10. #include <unistd.h>
11. #include <sys/types.h>
12.
13. #define PRINT_USEAGE  { \
14.    fprintf(stderr,"\n Usage: %s usec ",argv[0]); \
15.    fprintf(stderr,"\n\n");\
16.   }
17.
18. int main (int argc, char **argv)
19. {
20.   unsigned int nTimeTestSec = 0;         /* sec */
21.   unsigned int nTimeTest = 0;          /* usec */
22.   struct timeval tvBegin;
23.   struct timeval tvNow;
24.   int ret = 0;
25.   unsigned int nDelay = 0;          /* usec */
26.   fd_set rfds;
27.   struct timeval tv;
28.   int fd = 1;
29.   int i = 0;
30.   struct timespec req;
31.   unsigned int delay[20] =
32.     { 500000, 100000, 50000, 10000, 1000, 900, 500, 100, 10, 1, 0 };
33.   int nReduce = 0;                  /* 误差  */
34.
35. #if 0
36.   if (argc < 2)
37.     {
38.       PRINT_USEAGE;
39.       exit (1);
40.     }
41.   nDelay = atoi (argv[1]);
42. #endif
43.
44.
  fprintf (stderr, "%18s%12s%12s%12s\n", "function", "time(usec)", "realTi
me",
45.            "reduce");
46.   fprintf (stderr,
```

```
47.            "-------------------------------------------------------------
---------\n");
48.
49.   for (i = 0; i < 11; i++)
50.     {
51.       if (delay[i] < 0)
52.         break;
53.       nDelay = delay[i];
54.
55.       /*      test usleep */
56.       gettimeofday (&tvBegin, NULL);
57.       ret = usleep (nDelay);
58.       if (-1 == ret)
59.         {
60.           fprintf (stderr, " usleep error . errno=%d [%s]\n", errno,
61.                    strerror (errno));
62.         }
63.       gettimeofday (&tvNow, NULL);
64.       nTimeTest =
65.         (tvNow.tv_sec - tvBegin.tv_sec) * 1000000 + tvNow.tv_usec -
66.         tvBegin.tv_usec;
67.       nReduce = nTimeTest - nDelay;
68.
      fprintf (stderr, "/t usleep       %8u   %8u   %8d\n", nDelay, nTimeT
est,nReduce);
69.
70.
71.       /*      test nanosleep */
72.       gettimeofday (&tvBegin, NULL);
73.       req.tv_sec = nDelay / 1000000;
74.       req.tv_nsec = (nDelay % 1000000) * 1000;
75.       ret = nanosleep (&req, NULL);
76.       if (-1 == ret)
77.         {
78.
          fprintf (stderr, "/t nanosleep    %8u   not support\n", nDelay);
79.         }
80.       else
81.         {
82.           gettimeofday (&tvNow, NULL);
83.           nTimeTest =
84.
            (tvNow.tv_sec - tvBegin.tv_sec) * 1000000 + tvNow.tv_usec -
85.              tvBegin.tv_usec;
86.           nReduce = nTimeTest - nDelay;
87.
          fprintf (stderr, "/t nanosleep    %8u   %8u   %8d\n", nDelay,
88.                   nTimeTest, nReduce);
```

```
89.         }
90.
91.       /*      test select */
92.       gettimeofday (&tvBegin, NULL);
93.       FD_ZERO (&rfds);
94.       FD_SET (fd, &rfds);
95.       tv.tv_sec = 0;
96.       tv.tv_usec = nDelay;
97.       ret = select (0, NULL, NULL, NULL, &tv);
98.       if (-1 == ret)
99.         {
100.
          fprintf (stderr, " select error . errno=%d [%s]\n", errno,
101.                   strerror (errno));
102.         }
103.      gettimeofday (&tvNow, NULL);
104.      nTimeTest =
105.        (tvNow.tv_sec - tvBegin.tv_sec) * 1000000 + tvNow.tv_usec -
106.        tvBegin.tv_usec;
107.      nReduce = nTimeTest - nDelay;
108.
      fprintf (stderr, "/t select       %8u   %8u   %8d\n", nDelay, nTimeT
est,
109.                nReduce);
110.
111.    }
112.
113.  return 0;
114. }
```

程序显示如下：

[java] view plain copy

print?

```
1. [root@localhost test]# ./sleep_com
2.           function  time(usec)    realTime      reduce
3. ------------------------------------------------------------------
4. /t usleep          500000     501575       1575
5. /t nanosleep       500000     501861       1861
6. /t select          500000     499893       -107
7. /t usleep          100000     101933       1933
8. /t nanosleep       100000     101957       1957
9. /t select          100000      99946        -54
10. /t usleep           50000      51954       1954
11. /t nanosleep        50000      51962       1962
12. /t select           50000      49991         -9
13. /t usleep           10000      11941       1941
```

14.	/t nanosleep	10000	11973	1973
15.	/t select	10000	9974	-26
16.	/t usleep	1000	2976	1976
17.	/t nanosleep	1000	2974	1974
18.	/t select	1000	993	-7
19.	/t usleep	900	1968	1068
20.	/t nanosleep	900	1978	1078
21.	/t select	900	966	66
22.	/t usleep	500	1971	1471
23.	/t nanosleep	500	1973	1473
24.	/t select	500	992	492
25.	/t usleep	100	1970	1870
26.	/t nanosleep	100	1979	1879
27.	/t select	100	968	868
28.	/t usleep	10	1972	1962
29.	/t nanosleep	10	1974	1964
30.	/t select	10	993	983
31.	/t usleep	1	1969	1968
32.	/t nanosleep	1	1983	1982
33.	/t select	1	960	959
34.	/t usleep	0	988	988
35.	/t nanosleep	0	961	961
36.	/t select	0	5	5
37.	/t usleep	0	971	971

通过上表可以看出usleep(1000)实际 延时将近3ms。