

## # 功能要求

---

实现从数字1到数字N的累加

filename : sum.c, 可执行程序

sum Usage: ./sum -i 5 -n 10000 : 表示使用5个进程, 计算从1加到10000的和

Result:  $1+2+3+...+10000 = ?$

1. 每个进程都参与计算
2. 不能对业务进行拆分
3. 每个进程都去抢着加
4. 不能使用任何睡眠策略
5. 不能规定进程的执行顺序

## # 运行效果

---

```
+ syscoding ./a.out -i 4 -n 100
proces is 4
max val is 100
Result is : 5050
+ syscoding ./a.out -i 4 -n 1000
proces is 4
max val is 1000
Result is : 500500
+ syscoding ./a.out -i 4 -n 10000
proces is 4
max val is 10000
Result is : 50005000
+ syscoding ./a.out -i 4 -n 100000
proces is 4
max val is 100000
Result is : 5000050000
+ syscoding ./a.out -i 4 -n 1000000
proces is 4
max val is 1000000
Result is : 500000500000
+ syscoding ./a.out -i 4
proces is 4
Result is : 500500
+ syscoding vim sum.c
+ syscoding ./a.out
Using default config: i = 1, n = 1000
Result is : 500500
```

## code

```
1  #include <fcntl.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <sys/file.h>
6  #include <sys/stat.h>
7  #include <sys/types.h>
8  #include <unistd.h>
9  #include <wait.h>
10
11 #define BUFLen 100
12
13 int main(int argc, char** argv) {
14     int      opt, nproc = 1;
15     long long maxval = 1000;
16     long long res = 0, curval = 0;
17     char      buf[BUFLen] = {0};
18     while ((opt = getopt(argc, argv, "i:n:")) != -1) {
19         switch (opt) {
20             case 'i':
21                 nproc = atoi(optarg);
22                 printf("proces is %d \n", nproc);
23                 break;
24             case 'n':
25                 maxval = atol(optarg);
26                 printf("max val is %lld \n", maxval);
27                 break;
28             default:
29                 fprintf(stderr, "Usage: %s -i [process] -n
30 [limit]\n", argv[0]);
31                 exit(1);
32         }
33     }
34     if (argc == 1) {
35         printf("Using default config: i = 1, n = 1000\n");
36     }
37     // create a new file for IPC
```

```
38     pid_t pid;
39     int fd = open("./fileipc", O_RDWR | O_CREAT | O_TRUNC,
0666);
40     if (fd < 0) {
41         perror("open");
42         exit(1);
43     }
44     write(fd, "0 1", BUFLLEN);
45
46     // fork for multi-process
47     for (int i = 1; i <= nproc; i++) {
48         pid = fork();
49         if (pid < 0) {
50             perror("fork");
51             exit(1);
52         }
53         if (!pid) break;
54     }
55
56     // parent main process logic
57     if (pid) {
58         for (int i = 0; i < nproc; i++) {
59             waitpid(0, NULL, 0);
60         }
61         lseek(fd, 0, SEEK_SET);
62         if (read(fd, buf, BUFLLEN) < 0) {
63             perror("read");
64             exit(1);
65         }
66
67         sscanf(buf, "%lld %lld", &res, &curval);
68         printf("Result is : %lld\n", res);
69
70         close(fd);
71     }
72     // child process logic using file lock IPC
73     else {
74         close(fd);
75         fd = open("./fileipc", O_RDWR);
76         while (1) {
77             if (flock(fd, LOCK_EX) != 0) {
78                 perror("flock");
79                 exit(2);
```

```
80         }
81
82         lseek(fd, 0, SEEK_SET);
83         if (read(fd, buf, BUFLen) < 0) {
84             perror("read");
85             exit(1);
86         }
87         sscanf(buf, "%lld %lld", &res, &curval);
88
89         if (curval > maxval) {
90             close(fd);
91             exit(0);
92         }
93
94         res += curval;
95         ++curval;
96         sprintf(buf, "%lld %lld", res, curval);
97
98         lseek(fd, 0, SEEK_SET);
99         if (write(fd, buf, BUFLen) < 0) {
100             perror("write");
101             exit(1);
102         }
103
104         if (flock(fd, LOCK_UN) < 0) {
105             perror("flock unlock");
106             exit(1);
107         }
108     }
109 }
110
111 return 0;
112 }
```