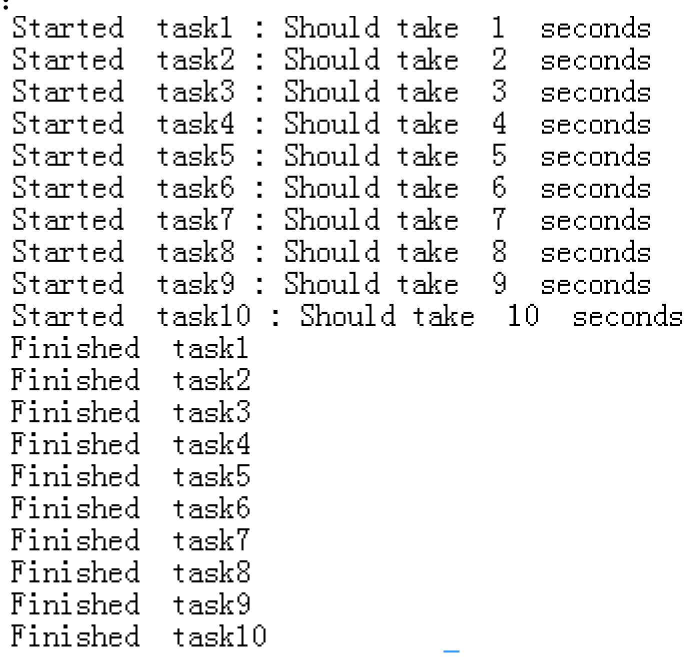
go 代码

```
/*
     File       : concurrent.go
     Author     : Mike
     E-Mail     : Mike_Zhang@live.com
*/

package main

import (
     "fmt"
     "time"
     "strconv"
)

func simulateEvent(name string,timeInSecs int64,ch chan
int){
     fmt.Println("Started ",name,": Should take
",timeInSecs," seconds")
     time.Sleep(time.Duration(timeInSecs) * time.Second)
     fmt.Println("Finished ",name)
     ch <- 1
}

func main(){
     var i int64
     var N int64
     N = 10
     ch := make(chan int)
     for i=1;i<=N;i++ {
          go
simulateEvent("task"+strconv.Itoa(int(i)),i,ch)
     }
     for i=1;i<=N;i++ {
          <-ch
     }
}
```

运行效果:



```
Started  task1 : Should take  1  seconds
Started  task2 : Should take  2  seconds
Started  task3 : Should take  3  seconds
Started  task4 : Should take  4  seconds
Started  task5 : Should take  5  seconds
Started  task6 : Should take  6  seconds
Started  task7 : Should take  7  seconds
Started  task8 : Should take  8  seconds
Started  task9 : Should take  9  seconds
Started  task10 : Should take  10  seconds
Finished  task1
Finished  task2
Finished  task3
Finished  task4
Finished  task5
Finished  task6
Finished  task7
Finished  task8
Finished  task9
Finished  task10           _
```

libevent 代码

```c
/*
     File       : concurrent.c
     Author     : Mike
     E-Mail     : Mike_Zhang@live.com
*/
/* gcc concurrent.c -o concurrent -levent */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <event2/event.h>
#include <event2/event_struct.h>

#define N 10
#define BUFLEN 256

struct timeval lasttime;

struct ST_EventWithDescription
{
   struct event *p_event;
   int time_interval;
   char lable[BUFLEN];
};

static void timeout_cb(evutil_socket_t fd, short event,
void *arg)
{
   struct timeval newtime, difference;
   struct ST_EventWithDescription *pSTEvent = arg;
   struct event *timeout = pSTEvent->p_event;
   double elapsed;

   evutil_gettimeofday(&newtime, NULL);
   evutil_timersub(&newtime, &lasttime, &difference);
   elapsed = difference.tv_sec + (difference.tv_usec /
1.0e6);

   printf("%s called at %d: %.3f seconds since my last
work.\n",
       (char*)pSTEvent->lable,(int)newtime.tv_sec,
elapsed);
   lasttime = newtime;

   struct timeval tv;
   evutil_timerclear(&tv);
   tv.tv_sec = pSTEvent->time_interval;
   event_add(timeout, &tv);
}

void setParam(struct ST_EventWithDescription
*stEventDescription,
         struct event *m_event,int time_interval,char*
m_lable)
{
   stEventDescription->p_event = m_event;
   stEventDescription->time_interval = time_interval;
   memset(stEventDescription-
>lable,0,sizeof(stEventDescription->lable));
   memcpy(stEventDescription-
>lable,m_lable,strlen(m_lable)+1);
}

void setTimeIntervalArr(int *arr,int n)
{
   int i;
   srand(time(NULL));
```

```
    for(i=0; i<n; ++i)
    {
     //    *(arr+i) = rand()%n + 1;
       *(arr+i) = i+1;
    }
}

int main(int argc, char **argv)
{
    struct event timeout[N];
    struct ST_EventWithDescription stEvent[N];
    int time_interval[N];
    int i=0;

    struct timeval tv;
    struct event_base *base;
    int flags = 0;

    setTimeIntervalArr(time_interval,N);

    base = event_base_new();
    evutil_timerclear(&tv);

    for(i=0; i<N; ++i)
    {
        char buf[BUFLEN]= {0};
        sprintf(buf,"task%d",i+1);
        setParam(stEvent+i,timeout+i,time_interval[i],buf);
        event_assign(timeout+i, base, -1, flags, timeout_cb,
(void*)(stEvent+i));
        event_add(timeout+i, &tv);
    }

    evutil_gettimeofday(&lasttime, NULL);
    event_base_dispatch(base);

    return (0);
}
```