TRƯỜNG ĐẠI HỌC BÁCH KHOA – ĐẠI HỌC QUỐC GIA TP.HCM
TRUNG TÂM KỸ THUẬT ĐIỆN TOÁN
268 Lý Thường Kiệt, Phường 14, Quận 10, TP.Hồ Chí Minh
Điện thoại: 84-8-3864 7256   ext 5371 – Fax: 84-8-3865 8687
Website: **www.cce.hcmut.edu.vn**
E-mail: **dientoan@hcmut.edu.vn**

Lab 7 - Homework

1 Homework

PROBLEM 1

Design and implement a logger support the two operations wrlog() and flushlog()$to manipulate the log data buffer "logbuf"

```
#define MAX_LOG_LENGTH 10
#define MAX_BUFFER_SLOT 6
#define MAX_LOOPS 30

string logbuf[MAX_BUFFER_SLOT];

int wrlog(char** logbuf, char* new_data);
int flushlog(char** logbuf);
```

For simplicity, we assume the buffer contains 6 (=MAX_BUFFER_SLOT) data slots and the flushlog() event occured when a predefined timeout expire. We also assume a LOG is a fixed length string, i.e. MAX_LOG_LENGTH

The behavious of the system can be illustrated as a sequence of write log data and the flush will be periodically trigger when a timeout is reached.

```
int main()
{
  wrlog(data1);
  wrlog(data2);
  wrlog(data3);
  ...
  wrlog(datan);
}
```

Further development: The interruptable mechanism of flush log can further support more unpredict event, i.e. it acknowledge a signall $SIGUSR1$, $SIGUSR2$ which are introduced in previous lab appendix.

GUIDE

You have been provided a non-protected implementation of logbuf program. This implementation includes the timing trigger flushlog() setup and a simple wrlog() instance. There are fixed MAX_LOOPS of thread calling wrlog(), and the behavior of the original progam as following illustration.

```
$ ./logbuf
flushlog()
wrlog(): 6
wrlog(): 7
wrlog(): 8
wrlog(): 9
wrlog(): 10
wrlog(): 11
wrlog(): 3
wrlog(): 13
wrlog(): 14
wrlog(): 15
wrlog(): 16
wrlog(): 17
wrlog(): 18
wrlog(): 2
wrlog(): 20
wrlog(): 21
wrlog(): 22
wrlog(): 23
wrlog(): 24
wrlog(): 25
wrlog(): 26
wrlog(): 27
wrlog(): 28
wrlog(): 29
wrlog(): 0
wrlog(): 5
wrlog(): 4
wrlog(): 12
wrlog(): 19
wrlog(): 1
flushlog()
flushlog()
flushlog()
flushlog()
flushlog()
flushlog()
flushlog()
```

The detailed implementation is include in this code listing.

```
#include <chrono>
#include <thread>
#include <iostream>
#include <pthread.h>
#include <unistd.h>

using namespace std;

#define MAX_LOG_LENGTH 10
#define MAX_BUFFER_SLOT 6
#define MAX_LOOPS 30

```

```
string logbuf[MAX_BUFFER_SLOT];

int count;

void* wrlog(void *data)
{
    int id = *(int *) data;

    usleep(20);
    cout << "wrlog(): " << id << "\n";fflush(stdout);

    return 0;
}


void flushlog()
{
    int i;

    cout << "flushlog()\n";

    for (i = 0; i < count; i++) {
       std::cout << "Slot " << i << ": " << logbuf[i] << "\n";
       logbuf[i] = std::to_string(-1);
    }
    fflush(stdout);

    /* Reset buffer */
    count = 0;

    return;

}

void timer_start(std::function<void(void)> func, unsigned int interval)
{
    std::thread([func, interval]() {
        while (true)
        {
            func();

std::this_thread::sleep_for(std::chrono::milliseconds(interval));
        }
    }).detach();
}

int main()
{
   int i;
   count = 0;
```

```
  pthread_t tid[MAX_LOOPS];
  int id[MAX_LOOPS];

  /* Setup periodically invoke flushlog() */
  timer_start(flushlog, 50);

  /* Asynchronous invoke task writelog */
  for (i = 0; i < MAX_LOOPS; i++) {
    id[i] = i;
    pthread_create(&tid[i], NULL, wrlog, (void *) &id[i]);
  }

  for (i = 0; i < MAX_LOOPS; i++)
    pthread_join(tid[i], NULL);

  sleep(5);

  return 0;
}
```

To compile the program, using this command (with the compiler support C++11 Stardard):

```
$ g++ -pthread -std=c++11 -o logbuf logbuf.cpp
```

Implement the protection mechanism to make it safe data access of wrlog() and flushlog() routines. If it has a proper configuration then the program behavious is somehow like this illustation

```
./logbuf
Slot 0: 5
Slot 1: 6
Slot 2: 7
Slot 3: 8
Slot 4: 9
Slot 5: 4
Slot 0: 10
Slot 1: 11
Slot 2: 12
Slot 3: 13
Slot 4: 14
Slot 5: 15
Slot 0: 16
Slot 1: 17
Slot 2: 18
Slot 3: 3
Slot 4: 19
Slot 5: 20
Slot 0: 21
Slot 1: 22
Slot 2: 23
Slot 3: 24
Slot 4: 25
Slot 5: 26
Slot 0: 27
Slot 1: 28
Slot 2: 2
Slot 3: 29
Slot 4: 1
Slot 5: 0
```