실습 3

Minsoo Ryu

Operating Systems and Distributed Computing Lab.
Hanyang University

msryu@hanyang.ac.kr

목차

- 실습
  - **Random Number Generation in Arbitrary Range**
  - **Multithread Monte Carlo Simulation for π**
  - **Shared Buffer**
  - **POSIX Shared Memory**

1. Random Number Generation (1)

❒ 구현 목표

- 주어지는 임의의 실수 구간 **[a, b]**에 속하는 랜덤값을 생성하여 화면에 출력하는 프로그램을 구현한다

❒ 구현 조건

- 아래의 함수를 구현한다

```
void generateRandBetween(float a, float b, int number, float result[])
```

  - Number는 생성될 랜덤값들의 개수, result[]는 생성된 랜덤값을 반환할 배열을 의미한다
  - Srand()와 rand() 함수만을 사용한다

❒ 제공 자료

- **generateRandBetween()**의 구현이 삭제된 **“rand_between.src”** 파일

1. Random Number Generation (2)

❒ 참고

- **int rand(void)**
  - [0, RAND_MAX] 구간에서 랜덤하게 정수값을 선택하여 반환한다
- **void srand(unsigned int seed)**
  - rand() 함수에 사용될 seed 값을 설정한다 (seed는 임의로 선택 가능)

❒ 실행 결과 예시

```
msryu@DESKTOP-AMQO5V8:~/Lab-Host/31_Random_Number_Generator$ ./executable
Enter a start value of range: -1.5
Enter a finish value of range: 2.5
Enter the total number of random values to generate: 1000

Random values in [-1.500000, 2.500000]
-0.212650 0.283018 1.700967 1.142708 -1.095587 -0.343805 2.103542 0.605036 2.226060 -0.295457 0.862718 -1.30
1587 1.661630 -0.703697 -0.542232 0.898232 1.860154 -0.942136 0.374052 -0.393001 -0.927306 1.555448 2.204424
 2.145574 -0.176428 2.157375 1.728721 0.483464 -0.113750 -0.782135 2.356474 1.173600 1.000882 1.557441 -0.18
3692 1.405296 -1.286364 -0.580150 -0.489668 2.439696 0.624393 1.873050 -1.361891 -0.213977 -1.330646 -0.4041
```

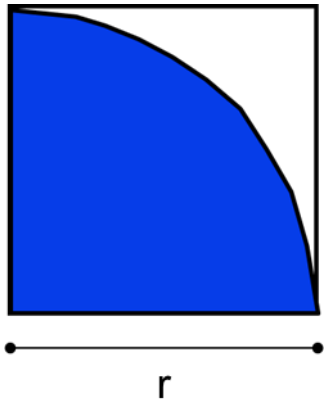
...

```
9 0.897435 -0.142504 -0.185222 2.443091 0.772485 0.536011 -0.033004 2.431211 2.217472 1.103367 -1.073491 -0.
379327 -0.785270 -1.162837 2.256492 1.664529 0.774831 -0.566719 1.856319 1.309133 -0.266741 -1.017578 0.1906
77 0.264325 1.699921 0.422913 -1.145085 -1.129870 2.221362 2.230873 0.309601 0.618797 -0.411631 1.624378 0.5
61888
The average of random values is 0.471908
```

2. Monte Carlo Simulation for π (1)

❒ 몬테카를로 시뮬레이션을 이용한 원주율 계산

- 가로와 세로의 길이가 각각 **2**인 정사각형에 랜덤하게 화살을 쏜다
  - 화살의 좌표를 (x, y)라 하면 $-1 \leq x \leq 1, -1 \leq y \leq 1$
- 반지름이 **r**인 원**(circle)** 내부에 화살이 맞은 횟수를 센다
  - 화살이 원의 내부에 맞을 조건은 $x^2 + y^2 \leq 1$
- 원과 정사각형의 면적의 비율은 $\pi : 4$
- 쏜 화살의 총 개수를 n, 원에 맞은 화살의 개수를 k라 하면 $\pi = 4 \times \frac{k}{n}$



2. Monte Carlo Simulation for π (2)

- 구현 목표
  - 멀티쓰레드로 **Monte Carlo** 시뮬레이션 방법을 사용하여 원주율 π 값을 근사적으로 계산하는 프로그램을 구현한다

- 구현 조건
  - **4**개의 쓰레드를 생성한다
  - 쓰레드로 실행될 함수는 아래의 형태를 가진다

```
void* simulateMonteCarlo(void* quadrant)
```

  - 아래의 **4**개의 함수를 사용한다
    - pthread_create()
    - pthread_join()
    - pthread_exit()
    - rand(): 화살이 맞는 좌표를 랜덤하게 생성하기 위해 사용
  - 쓰레드는 인자로 받은 ***quadrant**에 해당하는 사분면에 맞은 화살의 개수를 센다
  - 이전에 구현한 **generateRandBetween()** 함수를 사용한다 **(**별도의 **.c** 파일에 포함시키고**, Makefile**을 사용**)**

2. Monte Carlo Simulation for π (3)

❒ 제공 자료

- **simulateMonteCarlo()**의 구현이 삭제된 **“monte_carlo.src”** 파일
- **“rand_between.h”** 파일 및 **Makefile**

❒ 실행 결과 예시

- 쓰레드별 **100,000**회 시행 **(**총 **400,000**회**)**

```
msryu@DESKTOP-AMQO5V8:~/Lab-Host/32_Monte_Carlo_Simulation$ ./executable
Counts: 78567 78541 78589 78529
Sum of counts: 314226
The value of PI is 3.142260
```

3. Shared Buffer (1)

❒ 구현 목표

- 두 개의 쓰레드가 구조체형 데이터를 공유 버퍼를 이용하여 통신하는 프로그램을 구현한다

❒ 구현 조건

- 아래의 구조체와 공유 버퍼 배열을 사용한다

```
typedef struct {
    char name[16];
    float value;
} messageStruct;

messageStruct buffer[BUFFER_SIZE];
```

- 아래의 **pthread** 함수들을 사용한다
  - pthread_create(), pthread_join(), pthread_exit()

3. Shared Buffer (2)

- 구현 조건 (계속)
  - 아래와 같은 두개의 쓰레드 함수를 사용한다

```
void* producerThread(void* arg)
void* consumerThread(void* arg)
```

  - 아래의 뮤텍스 함수들을 사용한다

```
int pthread_mutex_lock(pthread_mutex_t *mutex)
int pthread_mutex_unlock(pthread_mutex_t *mutex)
```

- 제공 자료
  - **consumerThread** 함수의 구현이 삭제된 **shared_buffer.src**
  - 따라서 **consumerThread** 함수만 구현하면 되며, 제공되는 소스 코드의 **producerThread** 함수와 유사한 구조를 가지지만 대칭적으로 동작한다는 점을 고려해야 함

3. Shared Buffer (3)

❒ 실행 결과 예시

```
msryu@DESKTOP-AMQO5V8:~/Lab-Host/33_Shared_Buffer$ ./executable
Wrote Alice and 1 into slot 0.
Wrote Bob and 4 into slot 1.
Wrote Carol and 7 into slot 2.
Wrote David and 0 into slot 3.
Wrote Erwin and 3 into slot 4.
Wrote Alice and 1 into slot 5.
Wrote Bob and 4 into slot 6.
Read Alice and 1 from slot 0.
Wrote Carol and 7 into slot 7.
Wrote David and 0 into slot 8.
Wrote Erwin and 3 into slot 9.
Read Bob and 4 from slot 1.
Read Carol and 7 from slot 2.
Read David and 0 from slot 3.
Read Erwin and 3 from slot 4.
Read Alice and 1 from slot 5.
Read Bob and 4 from slot 6.
Read Carol and 7 from slot 7.
Read David and 0 from slot 8.
Read Erwin and 3 from slot 9.
```

4. POSIX Shared Memory (1)

❒ 구현 목표

- **POSIX Shared Memory**를 이용하여 통신하는 **2**개의 프로그램을 작성한다

❒ 구현 조건

- 이전 실습에서 사용한 메시지 구조체가 포함된 **msg_data.h** 파일을 사용한다

```
typedef struct {
    char name[16];
    float value;
} messageStruct;
```

- 아래와 같은 두개의 프로그램에서 각각 **msg_data.h**를 **include**한다
  - producer.c 및 consumer.c

4. POSIX Shared Memory (2)

- 구현 조건 (계속)
  - **“shared_memory_producer.c”**에서는 메시지 구조체에 기입된 데이터를 공유 메모리에 복사한다 **(**구조체 **5**개의 데이터**)**
  - **“shared_memory_consumer.c”**에서는 공유 메모리에 복사된 값들을 읽어 화면에 출력한다
  - **“shared_memory_producer.c”**에서는 아래의 함수들을 사용한다
    - shm_open(), ftruncate(), mmap(), strcpy(), memcpy(), munmap(), close()
  - **“shared_memory_consumer.c”** 에서는 아래의 함수들을 사용한다
    - shm_open(), mmap(), memcpy(), munmap(), close(), shm_unlink()
    - 또한 “shared_memory_producer.c”가 공유 메모리에 쓴 데이터를 읽어 화면에 출력한다

- 제공 자료
  - **“msg.h”, “shared_memory_producer.c”, “Makefile”**
  - 따라서 **“shared_memory_consumer.c”**만 작성하면 됨
    - shared_memory_producer.c와 유사한 구조를 가지지만 대칭적으로 동작한다는 점을 고려해야 함

4. POSIX Shared Memory (3)

❒ 실행 결과 예시

```
msryu@DESKTOP-AMQO5V8:~/Lab-Host/34_POSIX_Shared_Memory_Struct$ ./producer
Wrote Alice and 1.230000 into 0th area.
Wrote Bob and 4.560000 into 1th area.
Wrote Carol and 7.890000 into 2th area.
Wrote David and 0.120000 into 3th area.
Wrote Erwin and 3.450000 into 4th area.
msryu@DESKTOP-AMQO5V8:~/Lab-Host/34_POSIX_Shared_Memory_Struct$ ./consumer
Read Alice and 1.230000 from 0th area.
Read Bob and 4.560000 from 1th area.
Read Carol and 7.890000 from 2th area.
Read David and 0.120000 from 3th area.
Read Erwin and 3.450000 from 4th area.
```


Q
A
thank you!