광운대학교
KwangWoon University

2014년 2학기 **운영체제** Assignment 2

Synchronization

Dept. of Computer Engineering,
Kwangwoon Univ.

Contents

- **Assignment 2**
- **소스코드 설명**
- **Sample Result**
- **Appendix**
  - Thread
  - POSIX Thread (pthread)
  - 소스 코드 Compile

Assignment 2 (1/5)

- **POSIX Thread를 이용한 Readers-Writers Problem 해결**
  - Readers-Writers Problem
    - 강의자료 "**Chap. 5**" p.22 ~ p.23 참고

writer

```
while (true) {
        wait(wrt);

        // writing is performed

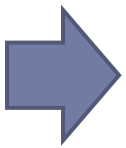
        signal(wrt);
}
```

reader

```
while (true) {
        wait(mutex);
        readcount++;
        if (readcount == 1)    wait(wrt);
        signal(mutex);

        // reading is performed

        wait(mutex);
        readcount--;
        if (readcount  == 0)   signal(wrt);
        signal(mutex);
}
```



pthread와 pthread의 mutex를 이용하여 구현

  - POSIX Thread (pthread)
    - Thread
      - 강의자료 "**Chap. 4**" 참고
    - pthread API
      - 본 자료의 "**Appendix**" 참고

Assignment 2 (2/5)

- **Report**
  - Introduction
  - Program Analysis in terms of **Synchronization**
    - 본인이 추가한 코드에 의한 프로그램 결과가 옳은 것인지를 그 내용이 잘 전달되도록 자유롭게 작성.
  - Reference
    - e.g. 친구 도움, 책, 인터넷 사이트 주소 등
    - 강의자료만 이용한 경우 생략 가능
  - Conclusion

- **Source**
  - 추가한 코드 라인 수의 50%이상 주석 작성
  - **Compile 에러 시 0점 처리**

- **Copy 발견 시 0점 처리**

Assignment 2 (3/5)

- **Submit**
  - Hardcopy
    - Report를 출력하여 **수업 시간 (11월 7일 금요일) 시작 전 제출**
  - Softcopy
    - **11월 6일 목요일 23시 59분 59초** 까지 **U-Campus에 제출**
    - Report와 소스코드, Makefile을 첨부하여 zip 파일로 압축하여 제출.
    - 압축 파일 이름 : Assignment2_학번.zip
      - E.g. Assignment2_2014123456.zip

Assignment 2 (4/5)

- **주어진 소스의 아래와 같은 주석 부분에 코드를 추가하여 문제를 해결한다.**
  - ```
    // ---
    // your codes
    // ---
    ```
  - 총 다섯 부분 ➔ Line 75~77, 81~83, 87~89, 100~102, 106~108
  - 해당 부분에 추가하는 라인 수는 주석 수와 무관함.
    - 즉, **주석이 세 줄이라고 해서 코드가 세 줄이라는 의미는 아님**.

- **소스코드를 완성하고, 본인이 작성한 코드가 올바르게 동작함을 보일 것.**
  - **자신이 작성한 코드와 그 동작 결과**에 대해, 본인의 프로그램 상의 Reader, Writer thread가 synchronization 문제를 일으키지 않고 **올바르게 동작함을 보이면 됨**.
    - 정상적으로 구현하였다면, 동작 결과는 수행할 때 마다 달라지므로 자신의 프로그램이 제대로 동작함을 보여주는 적절한 결과를 선택하여 사용하면 됨.
    ```

Assignment 2 (5/5)

- **아래의 환경에서 구현 및 테스트**
  - OS : Ubuntu 12.04 LTS 32bits (recommended – on **virtual machine**)
  - Compiler : GNU C Compiler (gcc)
    - **주어진 소스코드 Compile 방법은 Appendix 참고**

- **아래에서 언급하는 것 이외의 변수, 상수, 함수 등은 사용 불가**
  - 함수
    - POSIX Thread API in <pthread.h>
    - 해당 소스코드에 정의되어 있는 모든 함수
    - **rand( )** in <stdlib.h> ➔ **Writer가 값을 쓸 때, 해당 함수를 이용하여 무작위 값을 저장.**
  - 변수 ➔ 해당 소스코드에 정의되어 있는 모든 변수
  - 상수 ➔ 해당 소스코드에 정의되어 있는 모든 상수
  - 위의 사항을 지키지 않을 경우 해당 과제 **0점 처리**함

소스코드 설명 (1/5)

```
1  #include <stdio.h>
2     // for printf()
3  #include <pthread.h>
4     // for pthread~
5  #include <stdlib.h>
6     // for srand(), rand(), malloc(), free()
7  #include <time.h>
8     // for clock_gettime(), sleep(), struct timespec
9
10 #define NUM_OF_WRITER   3
11 #define NUM_OF_READER   10
12
13 #define TYPE_WRITER 0
14 #define TYPE_READER 1
15
16 pthread_mutex_t mutex;
17 pthread_mutex_t wrt;
18
19 int readcount = 0;
20
21 int shared_val;
22
23 void *pthread_reader(void *thread_num);
24 void *pthread_writer(void *thread_num);
25
26 void init();
27 void sleeping();
28
29 void msg_waiting        (int thread_type, int thread_num);
30 void msg_num_of_readers (int thread_num, int num_of_readers);
31 void msg_complete       (int thread_type, int thread_num, int written_data);
```

각종 API를 사용하기 위한 헤더파일 선언
(* 추가로 선언할 필요 없음)

Writer thread와 reader thread의 수.
(* 해당 값을 바꿔도 동작에 문제가 없도록 구현해야 함.)

Thread의 종류(writer, reader)를 구분하기 위해 사용
(* msg_waiting, msg_complete 함수의 첫 번째 인자)

Thread 간의 공유 변수에 관한 synchronization을 위해 사용
(* 강의자료 "Chap 5." 참고)

Reader thread 에서 사용 **(* 강의자료 "Chap 5." 참고)**

Reader가 읽고, writer가 쓰고자 하는 변수.

Reader, writer thread의 동작을 기술한 함수.

Pthread의 Mutex 초기화, Random seed 초기화, 공유 값 초기화 작업 수행

임의의 초(second) 동안 해당 함수를 부른 thread를 정지시킴

"--- .... ---- **Waiting**" 문자열 출력

"--- .... ---- **# of readers : ...**" 문자열 출력

읽고 쓴 값을 알려주는 문자열 출력

소스코드 설명 (2/5)

```
33 int main(void) {
34    pthread_t writer[NUM_OF_WRITER], reader[NUM_OF_READER];
35    void *writer_ret[NUM_OF_WRITER], *reader_ret[NUM_OF_READER];
36
37    int i;
38
39    init();
40
41    // Writter thread creation
42    for(i=0;i<NUM_OF_WRITER;i++) {
43       int *num = (int *)malloc(1 * sizeof(int));
44       *num = i;
45
46       pthread_create(&writer[i], NULL, pthread_writer, num);
47    }
48
49    // Reader thread creation
50    for(i=0;i<NUM_OF_READER;i++) {
51       int *num = (int *)malloc(1 * sizeof(int));
52       *num = i;
53
54       pthread_create(&reader[i], NULL, pthread_reader, num);
55    }
56
57    // Waiting for reader and writer threads to quit.
58    for(i=0;i<NUM_OF_WRITER;i++)  pthread_join(writer[i], &writer_ret[i]);
59    for(i=0;i<NUM_OF_READER;i++)  pthread_join(reader[i], &reader_ret[i]);
60
61    // Mutex destruction
62    pthread_mutex_destroy(&mutex);
63    pthread_mutex_destroy(&wrt);
64
65 }
```

Reader, writer thread 각각에 0번 부터 시작하는 고유의 번호 할당

소스코드 설명 (3/5)

```
67 void *pthread_reader(void *thread_num) {
68    int reader_num = *(int *)thread_num;
69    free((int *)thread_num);
70
71    sleeping();
72
73    msg_waiting(TYPE_READER, reader_num);
74
75    // ---
76    // your codes
77    // ---
78
79    msg_num_of_readers(reader_num, readcount);
80
81    // ---
82    // your codes
83    // ---
84
85    msg_complete(TYPE_READER, reader_num, shared_val);
86
87    // ---
88    // your codes
89    // ---
90 }
```

```
92 void *pthread_writer(void *thread_num) {
93    int writer_num = *(int *)thread_num;
94    free((int *)thread_num);
95
96    sleeping();
97
98    msg_waiting(TYPE_WRITER, writer_num);
99
100   // ---
101   // your codes
102   // ---
103
104   msg_complete(TYPE_WRITER, writer_num, shared_val);
105
106   // ---
107   // your codes
108   // ---
109 }
```

* 앞에서 언급한 바와 같이,

Writer에서 변수에 값을 저장할 때
rand() 함수로 부터 얻어온 무작위 값을 이용.

소스코드 설명 (4/5)

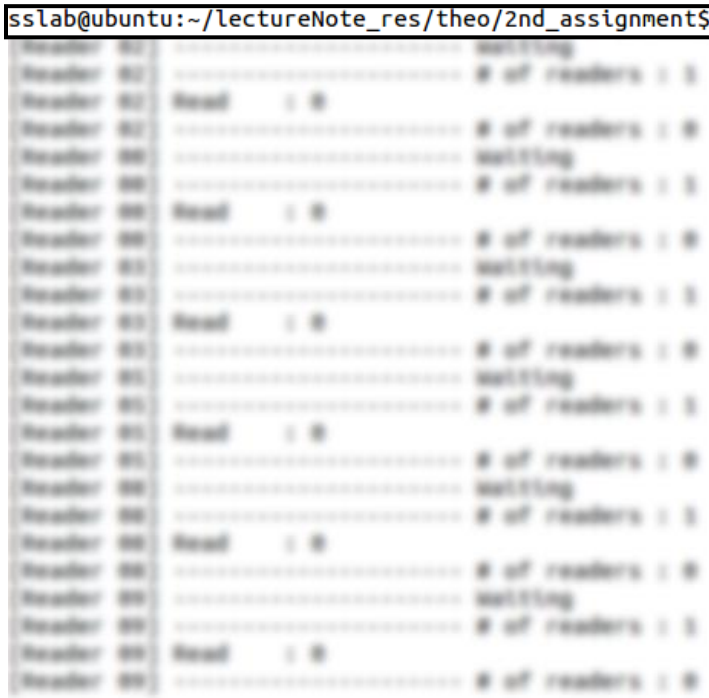
```
111 void init() {
112     struct timespec current_time;
113 
114     // Mutex initializaion
115     if(pthread_mutex_init(&mutex, NULL)) {
116         printf("mutex init error!\n");
117         exit(1);
118     }
119     if(pthread_mutex_init(&wrt, NULL)) {
120         printf("mutex init error!\n");
121         exit(1);
122     }
123 
124     // Random seed initializaion
125     clock_gettime(CLOCK_REALTIME, &current_time);
126     srand(current_time.tv_nsec);
127 
128     // Shared value initializaion
129     shared_val = 0;
130 }
131 
132 // Being slept during random for randomly selected time (< 1s)
133 void sleeping() {
134     sleep(1 + rand() % 2);
135 }
```

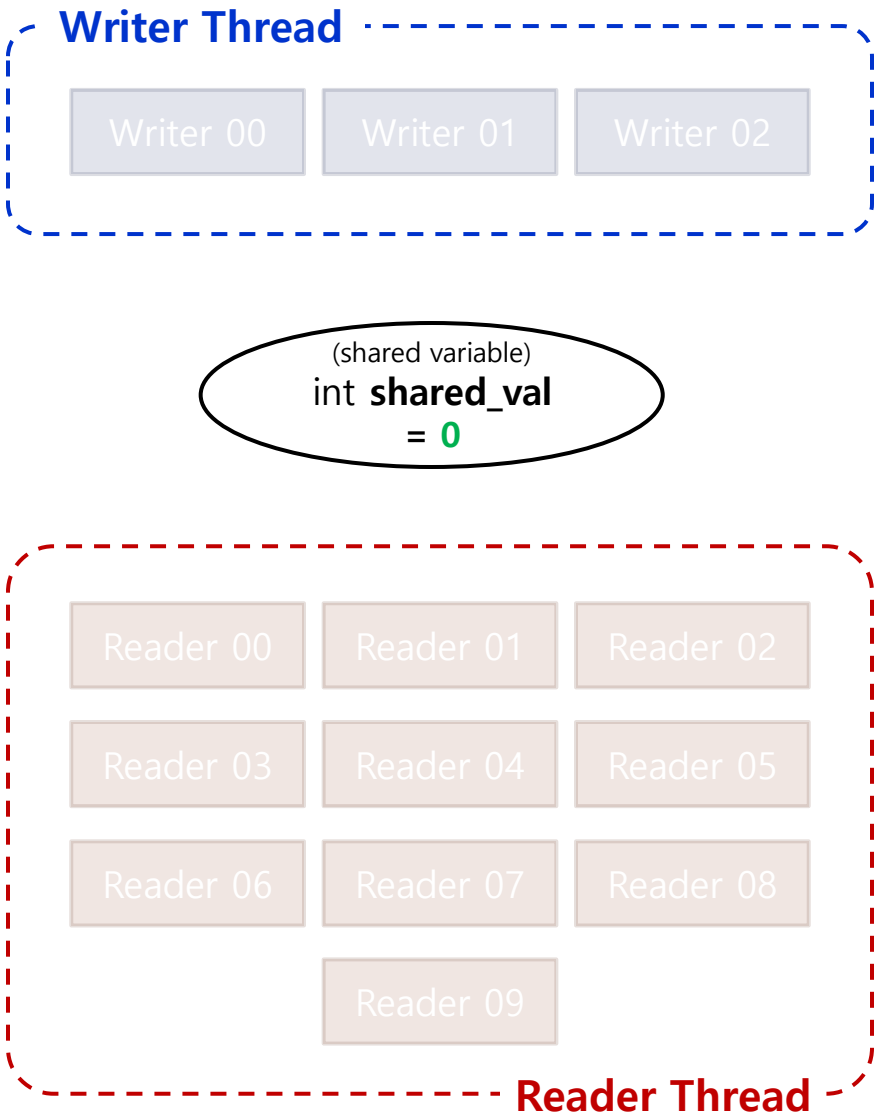
소스코드 설명 (5/5)

```
137 void msg_waiting(int thread_type, int thread_num) {
138    if(thread_type == TYPE_READER)
139       printf("[Reader %02d] -------------------- Waiting\n", thread_num);
140
141    else
142       printf("[Writer %02d] -------------------- Waiting\n", thread_num);
143 }
144
145 void msg_num_of_readers(int thread_num, int num_of_readers) {
146    printf("[Reader %02d] -------------------- # of readers : %d\n",
147          thread_num, num_of_readers);
148 }
149
150 void msg_complete(int thread_type, int thread_num, int written_data) {
151    if(thread_type == TYPE_READER)
152       printf("[Reader %02d] Read    : %d\n", thread_num, written_data);
153
154    else
155       printf("[Writer %02d] Written : %d\n", thread_num, written_data);
156 }
```

Sample Result (1/15)

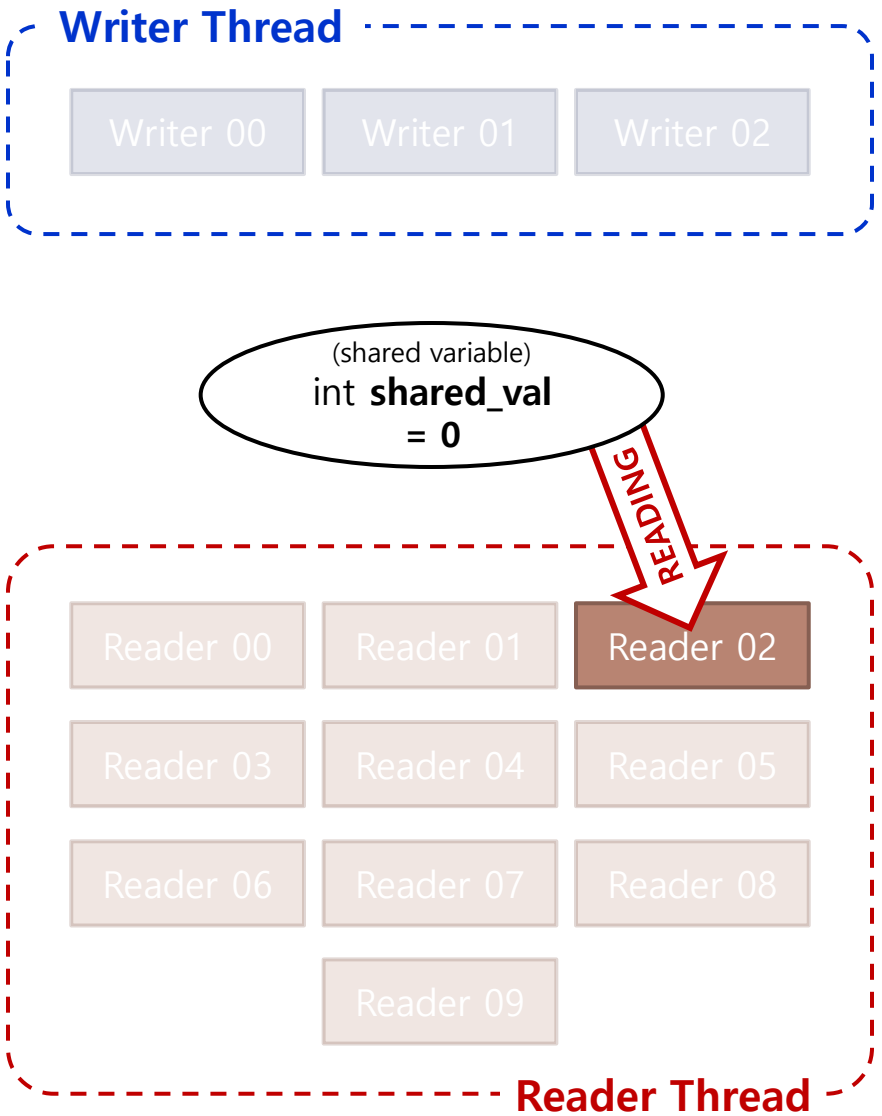
```
sslab@ubuntu:~/lectureNote_res/theo/2nd_assignment$ ./rw_problem
```





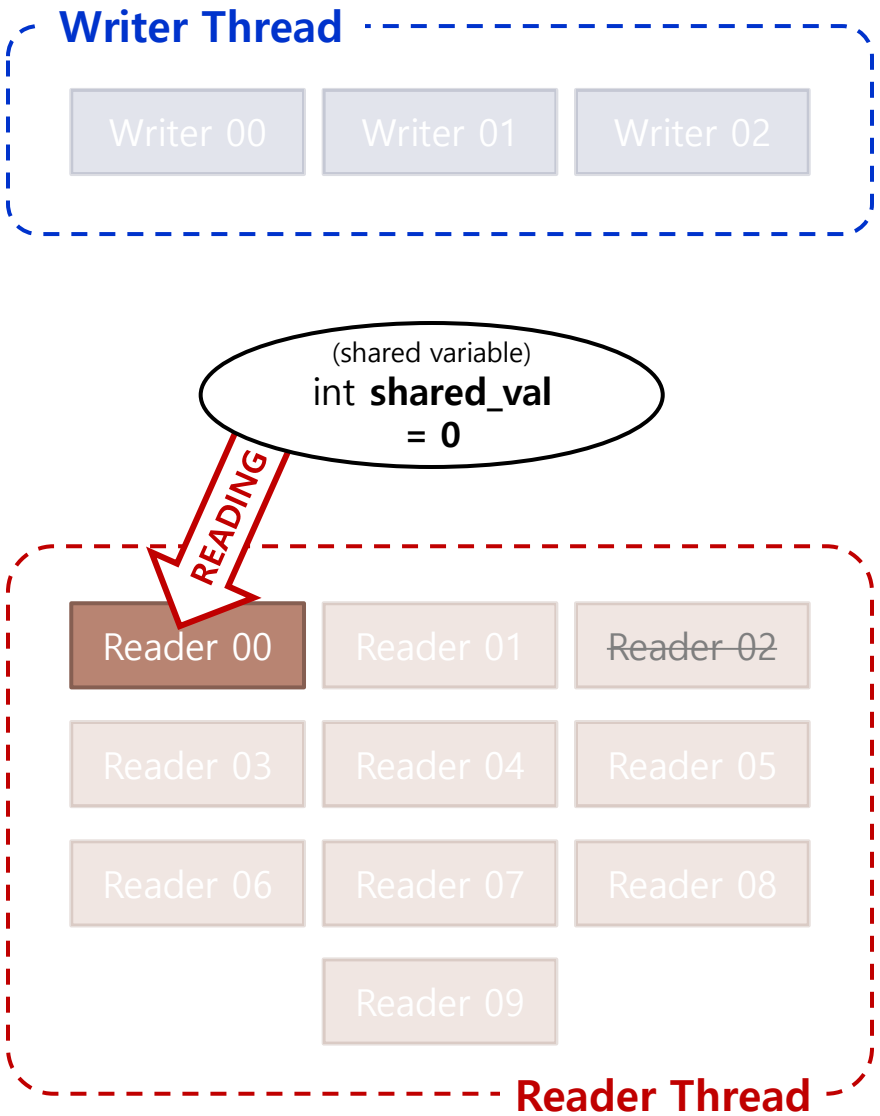
Sample Result (2/15)

```
[Reader 02] ---------------------- Waiting
[Reader 02] ---------------------- # of readers : 1
[Reader 02] Read    : 0
[Reader 02] ---------------------- # of readers : 0
```



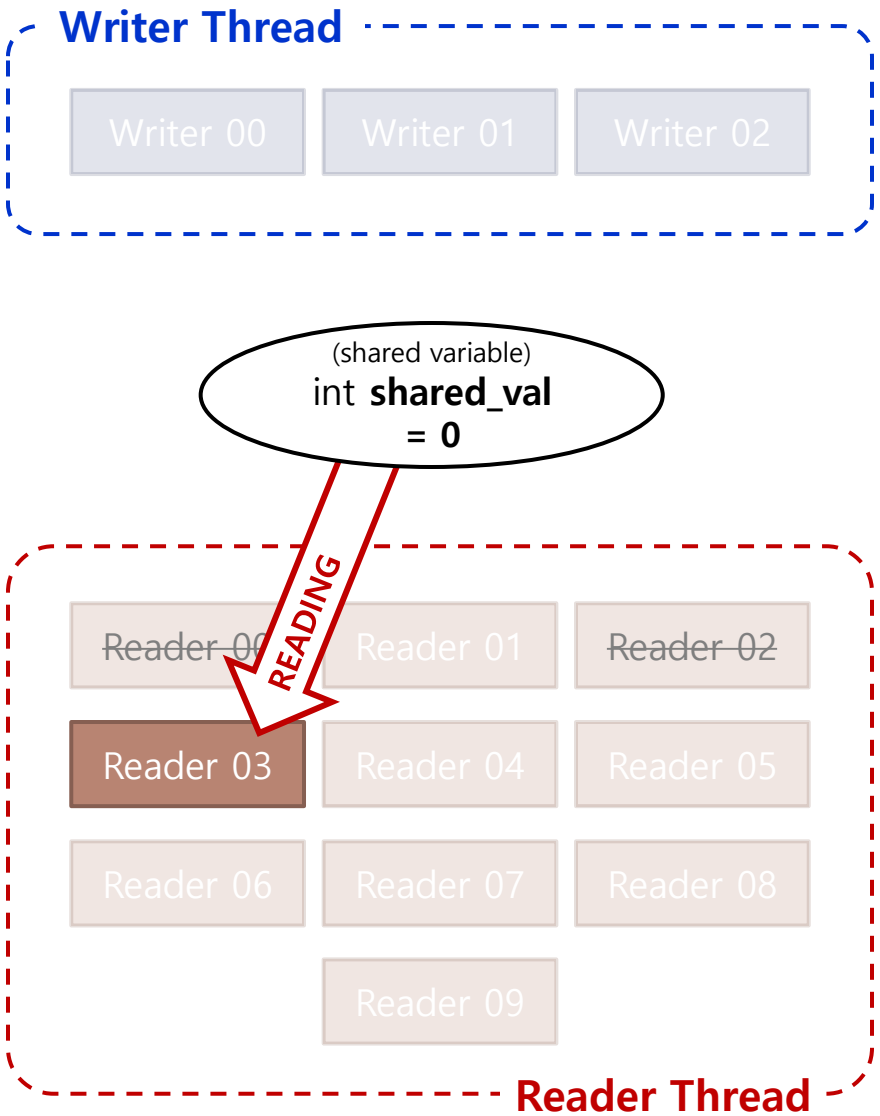
Sample Result (3/15)

```
[Reader 00] ---------------------- Waiting
[Reader 00] ---------------------- # of readers : 1
[Reader 00] Read     : 0
[Reader 00] ---------------------- # of readers : 0
```



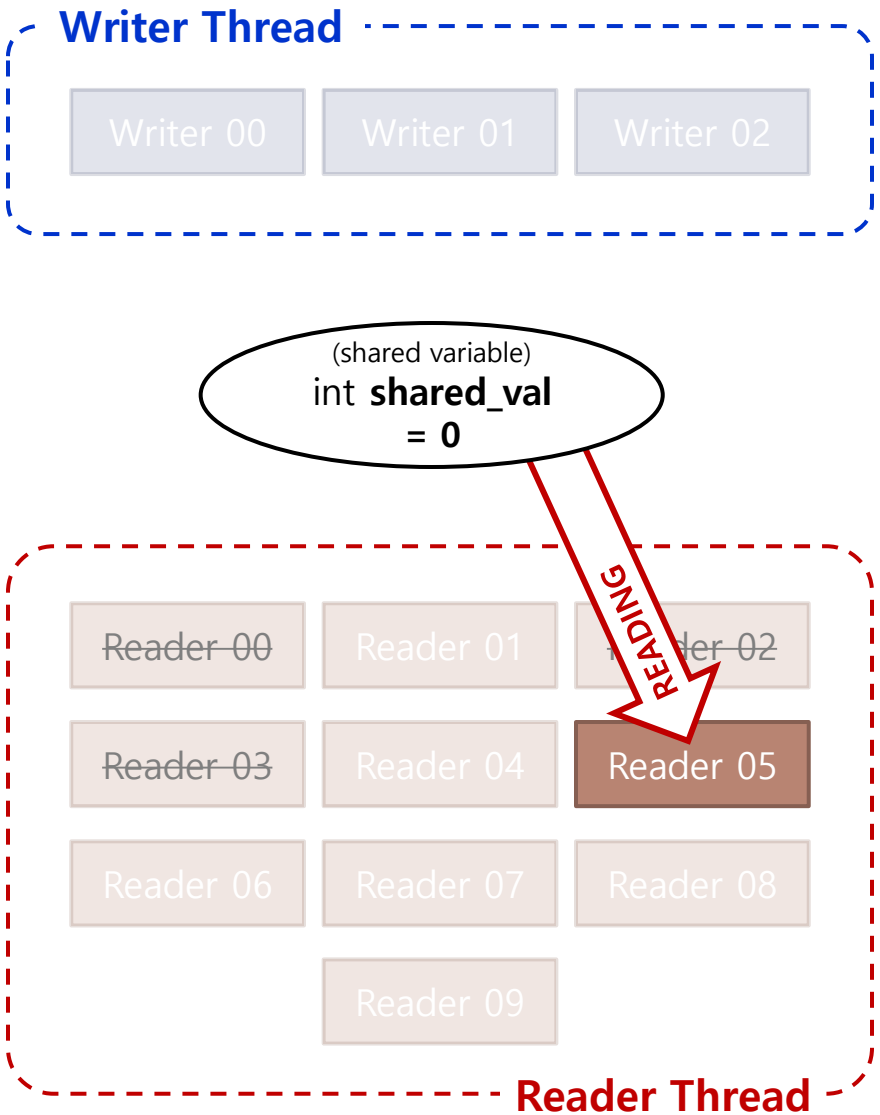
Sample Result (4/15)

```
[Reader 03] ---------------------- Waiting
[Reader 03] ---------------------- # of readers : 1
[Reader 03] Read     : 0
[Reader 03] ---------------------- # of readers : 0
```



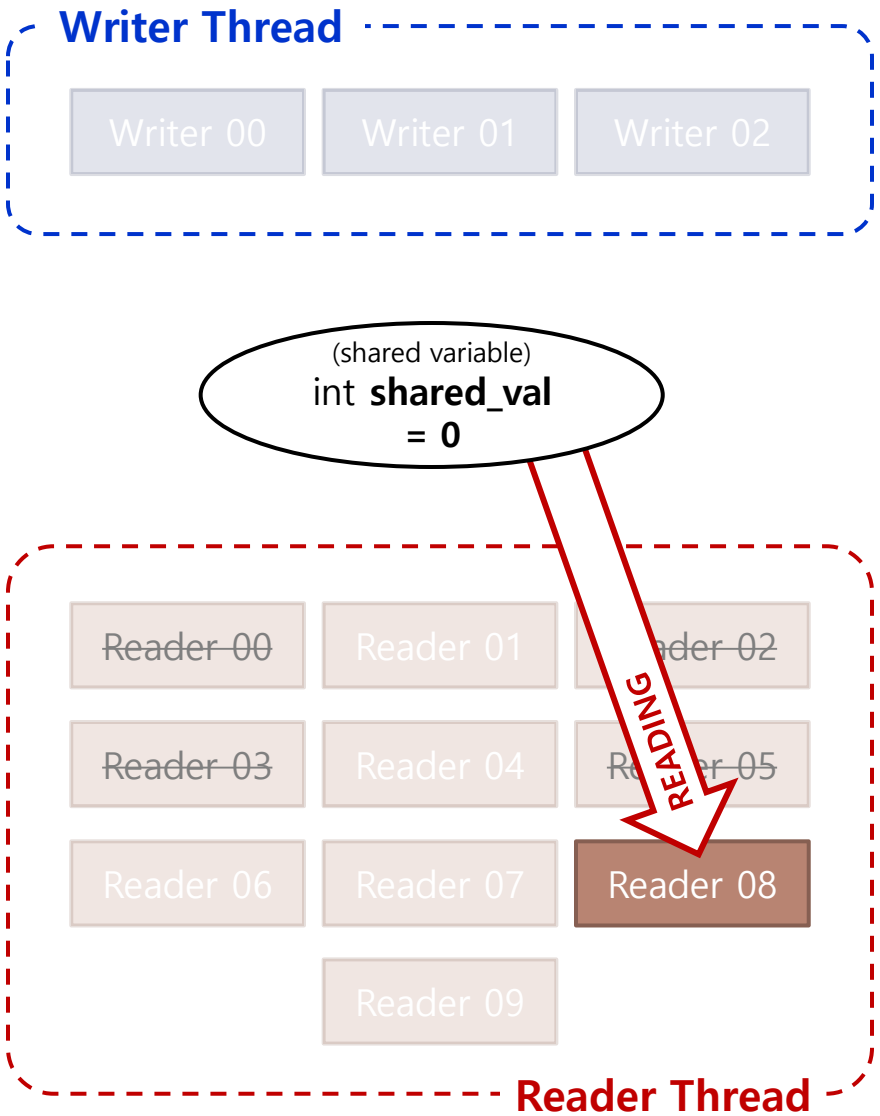
Sample Result (5/15)

```
[Reader 05] ---------------------- Waiting
[Reader 05] ---------------------- # of readers : 1
[Reader 05] Read     : 0
[Reader 05] ---------------------- # of readers : 0
```



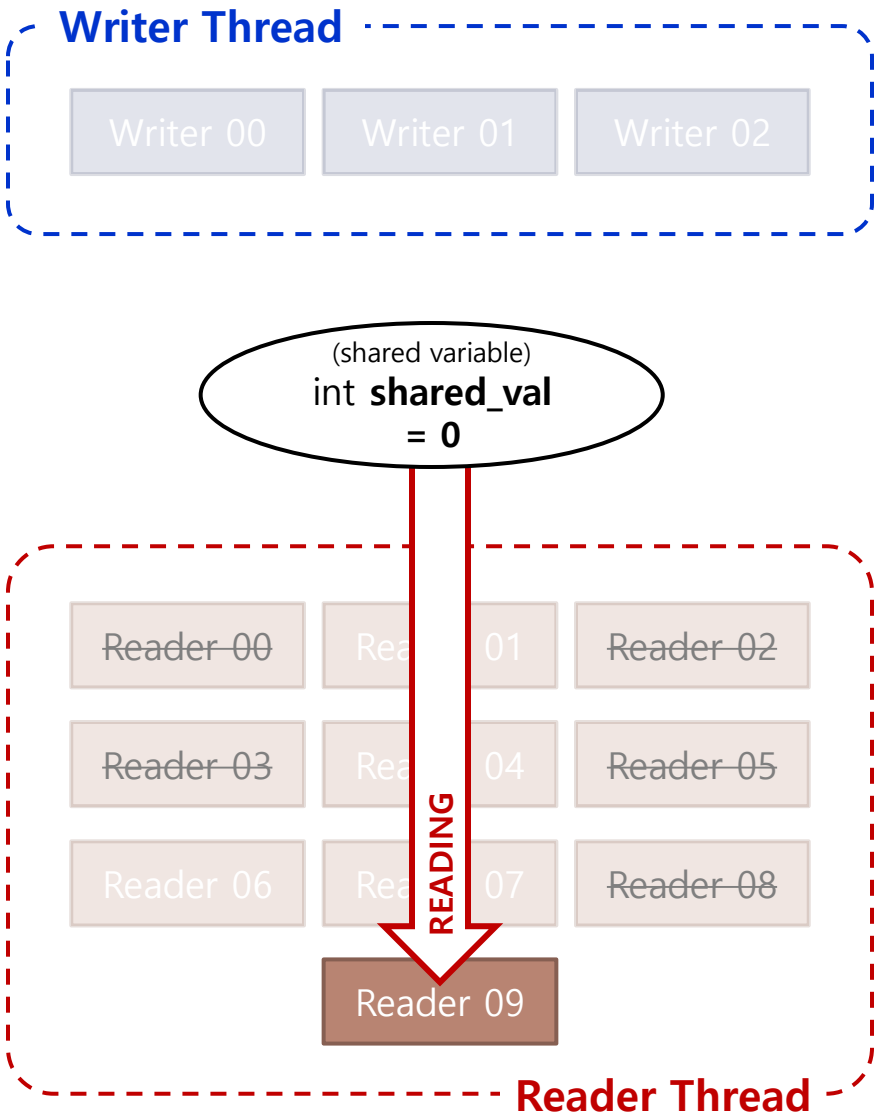
Sample Result (6/15)

```
[Reader 08] ---------------------- Waiting
[Reader 08] ---------------------- # of readers : 1
[Reader 08] Read     : 0
[Reader 08] ---------------------- # of readers : 0
```



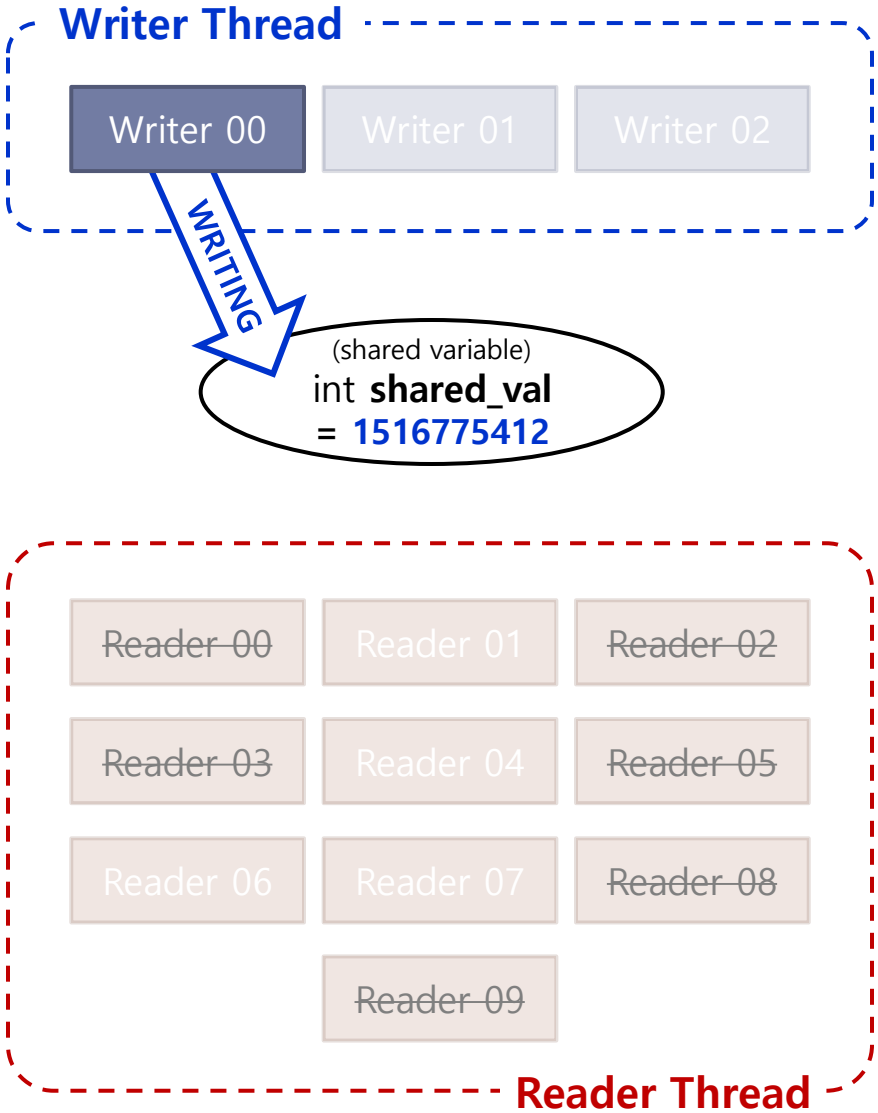
Sample Result (7/15)

```
[Reader 09] ---------------------- Waiting
[Reader 09] ---------------------- # of readers : 1
[Reader 09] Read     : 0
[Reader 09] ---------------------- # of readers : 0
```



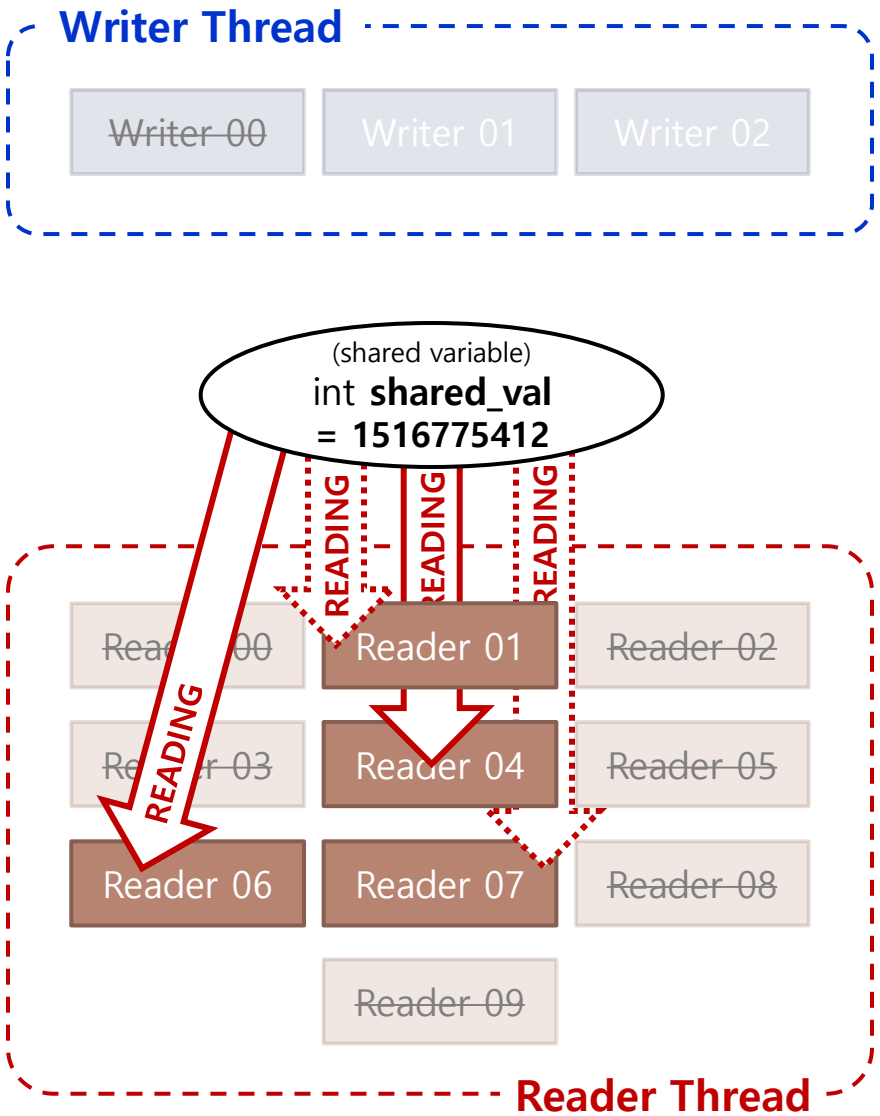
Sample Result (8/15)

```
[Writer 00] ---------------------- Waiting
[Writer 00] Written : 1516775412
```



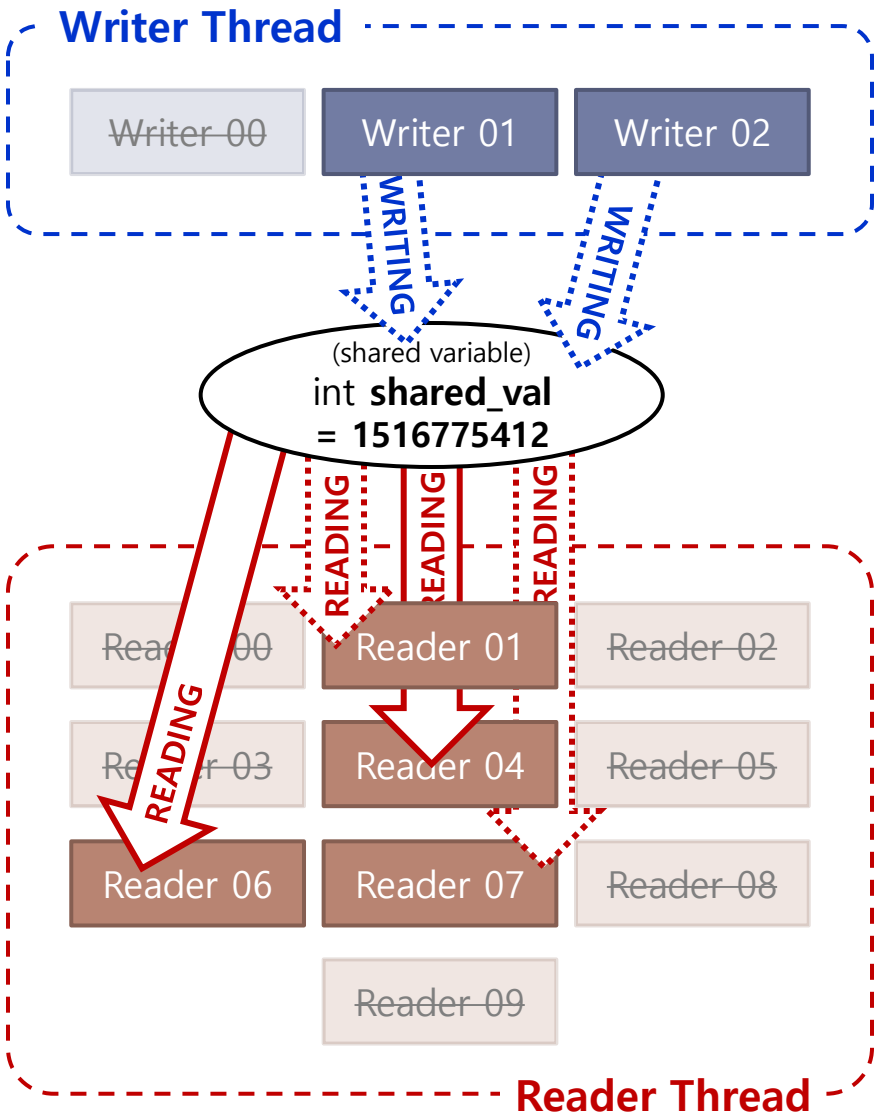
Sample Result (9/15)

```
[Reader 06] ---------------------- Waiting
[Reader 06] ---------------------- # of readers : 1
[Reader 07] ---------------------- Waiting
[Reader 04] ---------------------- Waiting
[Reader 04] ---------------------- # of readers : 2
[Reader 04] Read     : 1516775412
[Reader 06] Read     : 1516775412
[Reader 01] ---------------------- Waiting
```



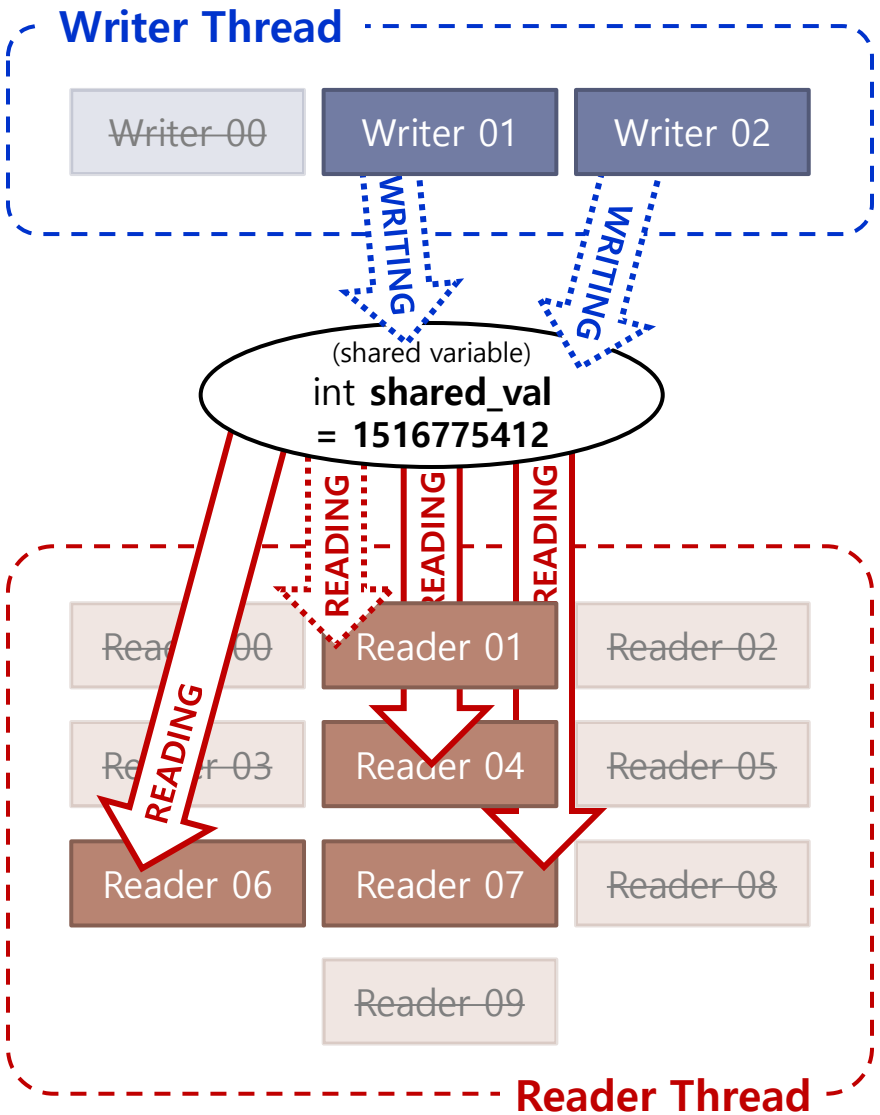
Sample Result (10/15)

```
[Writer 02] ---------------------- Waiting
[Writer 01] ---------------------- Waiting
```



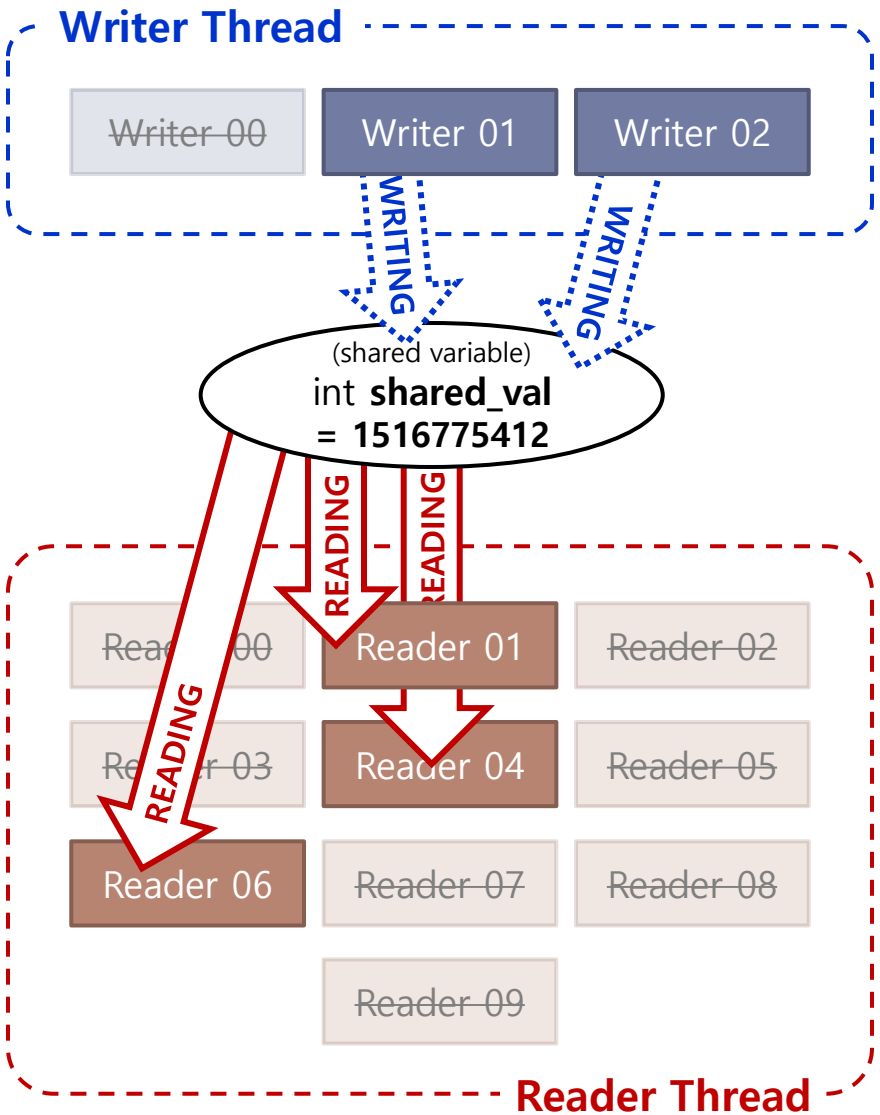
Sample Result (11/15)

```
[Reader 07] ---------- # of readers : 3
[Reader 07] Read     : 1516775412
[Reader 07] ---------- # of readers : 2
```

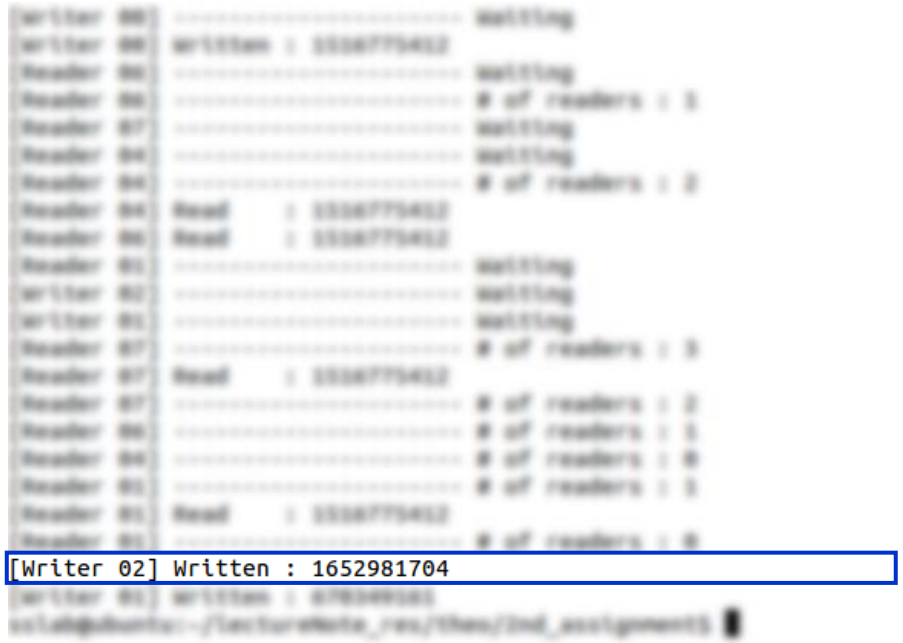


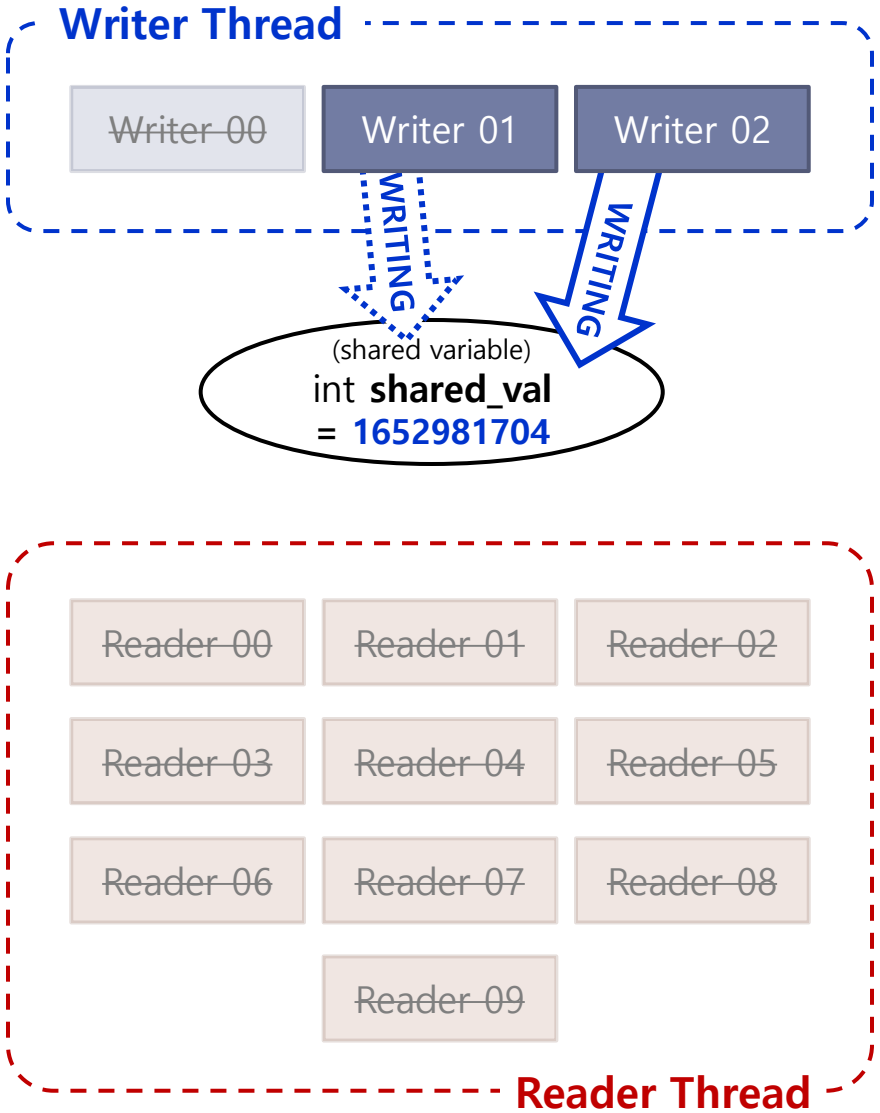
Sample Result (12/15)

```
[Reader 06] ---------------------- # of readers : 1
[Reader 04] ---------------------- # of readers : 0
[Reader 01] ---------------------- # of readers : 1
[Reader 01] Read     : 1516775412
[Reader 01] ---------------------- # of readers : 0
```



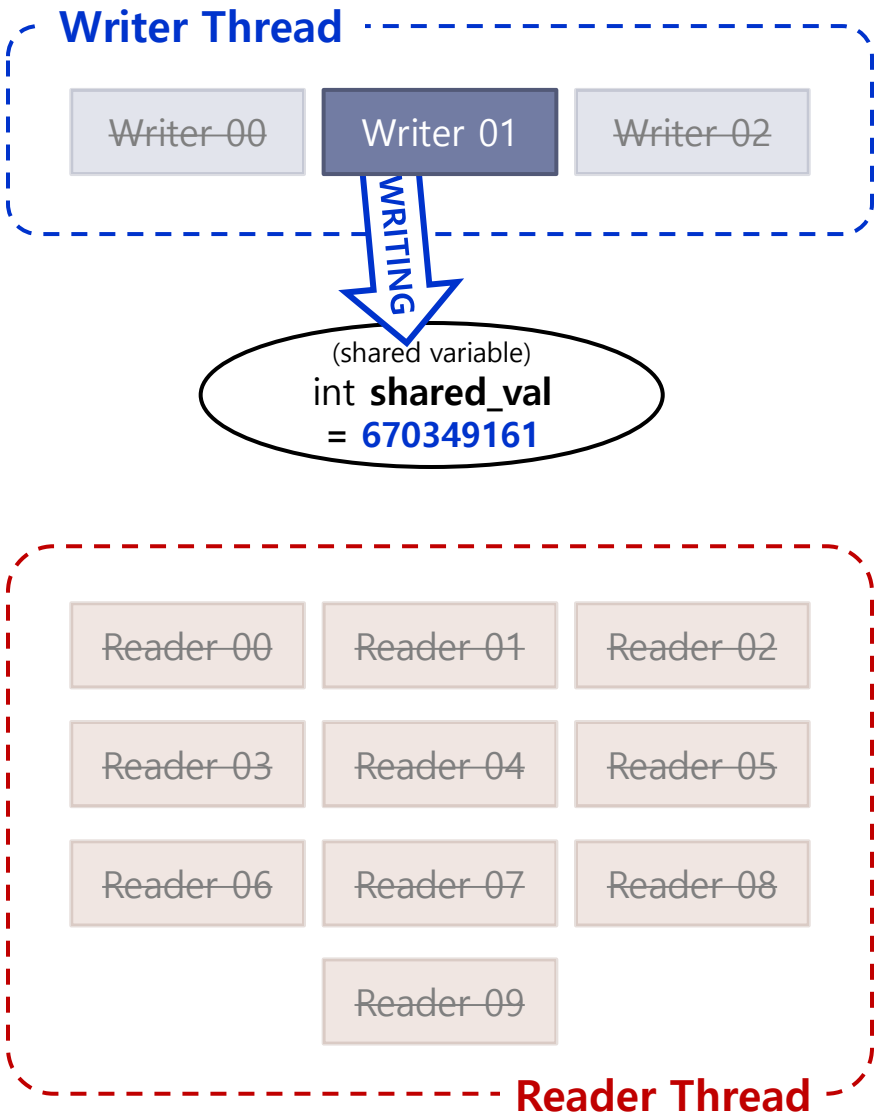
Sample Result (13/15)



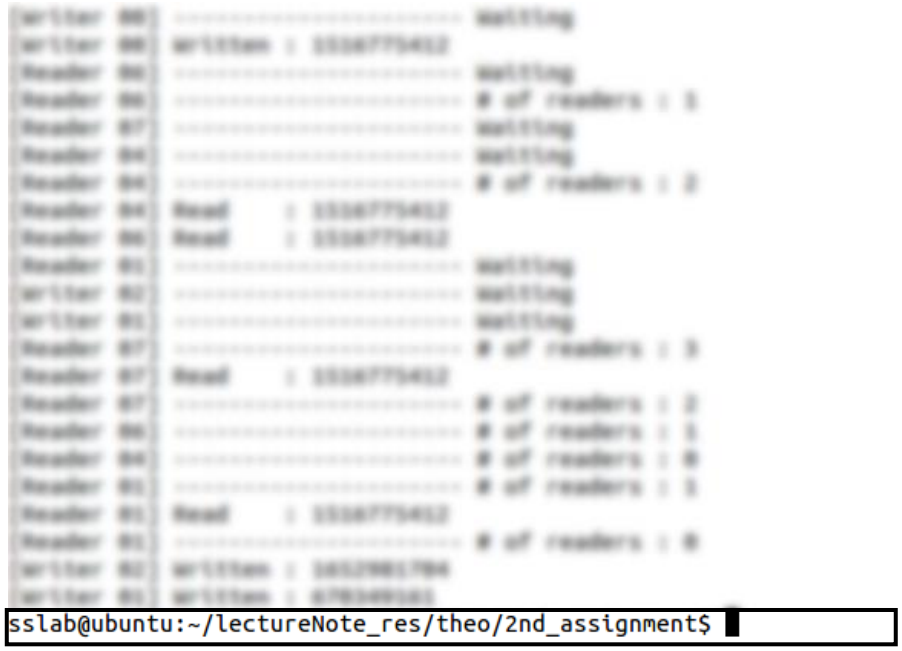


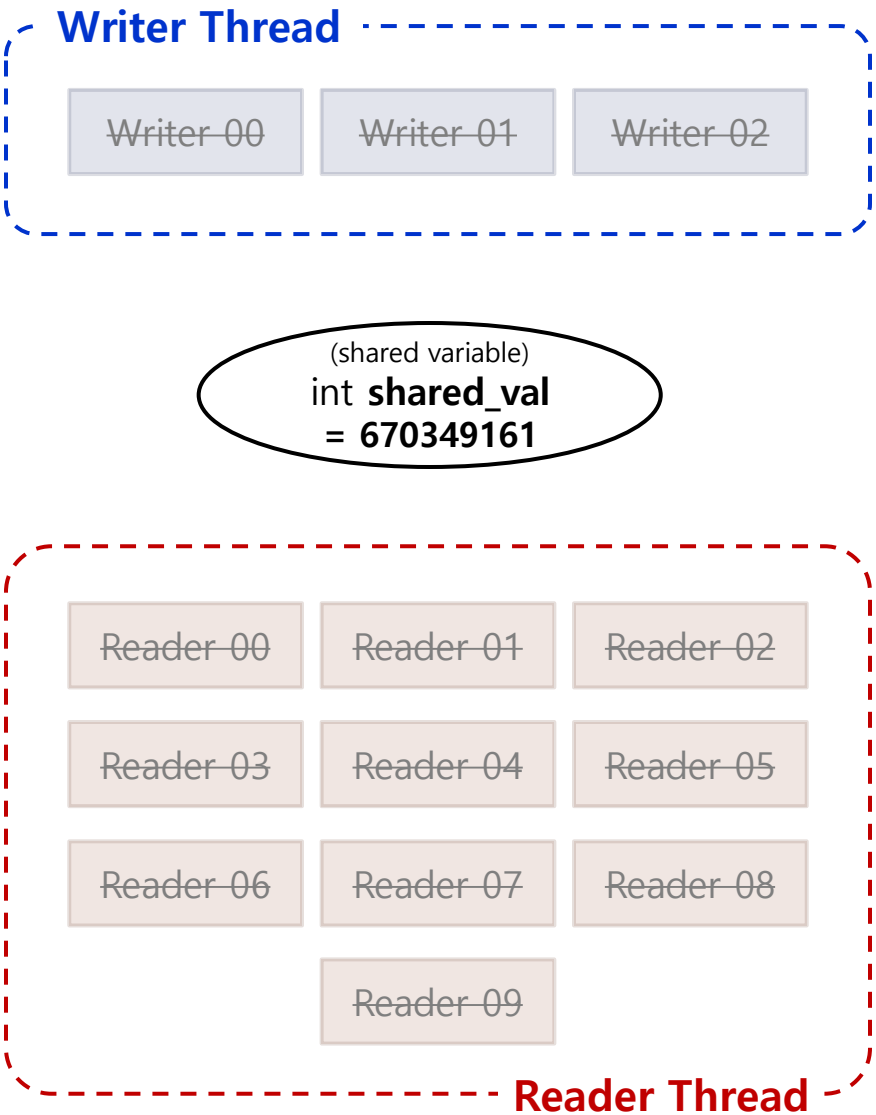
Sample Result (14/15)

```
[Writer 01] Written : 670349161
```



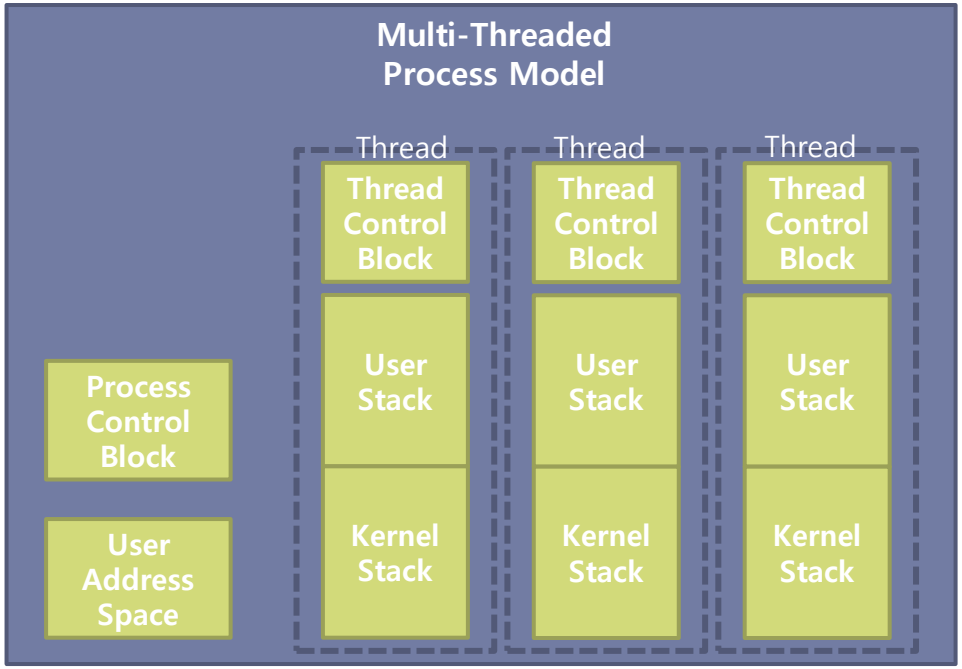
Sample Result (15/15)





Appendix (1/7)

- **Thread**
  - 특정 Process 내에서 실행되는 하나의 흐름을 나타내는 단위
  - 독립된 Program counter, Register Set, Stack을 가짐
  - 비 동기적인(Asynchronous) 두 개의 작업이 서로 독립적으로 진행 가능
  - 처리를 위해 조건 변수나 **Mutex**, Semaphore와 같은 방법을 사용함



Appendix (2/7)

- **POSIX**
  - 이식 가능 운영 체제 인터페이스 ("**P**ortable **O**perating **S**ystem **I**nterface")
  - 서로 다른 UNIX OS의 공통 API를 정리하여 이식성이 높은 유닉스 응용 프로그램을 개발하기 위한 목적으로 IEEE가 책정한 애플리케이션 인터페이스 규격

- **POSIX Thread (pthread)**
  - pthread_create()

```
#include <pthread.h>

int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start)(void*), void *arg);
```

  - pthread_t ***thread** : Thread ID를 의미하는 주소 값
  - const pthread_attr_t ***attr** : Thread 속성을 지정할 수 있는 값. 기본값은 NULL.
  - void *(***start**)(void*) : 특정 함수(**start**) 를 호출함으로써 thread가 시작됨.
  - void ***arg** : start 함수의 인자

Appendix (3/7)

- **POSIX Thread (pthread) (cont'd)**
  - pthread_join()
    - 지정된 thread가 종료될 때까지 호출 thread의 수행을 중단.

```
#include <pthread.h>

int pthread_join (pthread_t thread, void **value_ptr);
```

      - void ****value_ptr** : thread의 종료 코드가 저장될 장소.

    - waitpid()의 역할과 유사.

Appendix (4/7)

- **POSIX Thread (pthread) (cont'd)**
  - Mutex
    - "**Mut**ual **Ex**clusion"
    - Thread 간에 동시 접근을 허용하지 않겠다는 의미
    - Busy waiting 하며 대기함

    - pthread_mutex_init()

```
#include <pthread.h>

int pthread_mutex_init(pthread_mutex_t * mutex, const pthread_mutex_attr *attr);
```

      - pthread_mutex_t * **mutex** : 사용할 mutex 변수의 주소 값
      - const pthread_mutex_attr ***attr** : Mutex 속성 값. 기본 특징을 이용하고자 한다면, NULL.

    - pthread_mutex_lock() **// 다른 thread가 unlock하여 접근이 가능할 때 까지 대기함.**

```
#include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t *mutex);
```

      - pthread_mutex_t * **mutex** : 사용할 mutex 변수의 주소 값

Appendix (5/7)

- **POSIX Thread (pthread) (cont'd)**
  - Mutex
    - pthread_mutex_unlock()

```
#include <pthread.h>

int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

      - pthread_mutex_t * **mutex** : 사용할 mutex 변수의 주소 값

    - pthread_mutex_destroy()

```
#include <pthread.h>

int pthread_mutex_destroy(pthread_mutex_t *mutex);
```

      - pthread_mutex_t * **mutex** : 사용할 mutex 변수의 주소 값

Appendix (6/7)

- **주어진 소스 코드 컴파일**
  - 적당한 디렉토리에서, 첨부한 압축 파일을 푼다.
    - $ **tar –xvf 14-2_OSDes_Assignment02_Skeleton_Code.tar.gz**
  - 소스코드를 수정한다.
  - make 를 이용하여 컴파일 한다.
    - $ **make**
  - Error나 Warning이 없을 경우, **아래와 같이 출력**되며 "**rw_problem**"이 생긴다.

```
sslab@ubuntu:~/Desktop$ make
gcc -o rw_problem rw_problem.c -lpthread -lrt
sslab@ubuntu:~/Desktop$ ls
14-2_OSDes_Assignment02_Skeleton_Code.tar.gz  rw_problem
Makefile                                      rw_problem.c
sslab@ubuntu:~/Desktop$
```

    - -lpthread 옵션 ➔ **pthread**를 사용하기 위해 필요한 library 링크.
    - -lrt 옵션 ➔ **clock_gettime()** 함수를 사용하기 위해 필요한 library 링크.

Appendix (7/7)

- **주어진 소스 코드 컴파일 (cont'd)**
  - 생성된 rw_problem binary 파일을 실행하여 결과를 확인한다.

```
sslab@ubuntu:~/Desktop$ ./rw_problem
[Reader 01] -------------------- Waiting
[Reader 01] -------------------- # of readers : 0
[Reader 01] Read    : 0
[Writer 01] -------------------- Waiting
[Writer 01] Written : 0
[Reader 00] -------------------- Waiting
[Reader 00] -------------------- # of readers : 0
[Reader 00] Read    : 0
[Writer 00] -------------------- Waiting
[Writer 00] Written : 0
[Reader 04] -------------------- Waiting
[Reader 04] -------------------- # of readers : 0
[Reader 04] Read    : 0
[Reader 07] -------------------- Waiting
[Reader 07] -------------------- # of readers : 0
[Reader 07] Read    : 0
[Reader 08] -------------------- Waiting
[Reader 08] -------------------- # of readers : 0
[Reader 08] Read    : 0
[Reader 09] -------------------- Waiting
[Reader 09] -------------------- # of readers : 0
```