Producer and Consumer Problem Solution by. 12191579 김종하

Mutex를 이용한 Solution Pseudo Code

Variables

```
#define BUFFER_SIZE 30

int x = 0; // count, max 29
int in = 0;
int out = 0;
int buffer[BUFFER_SIZE];

pthread_mutex_t mutex;
pthread_cond_t buffer_has_space;
pthread_cond_t buffer_has_data;
```

Producer Thread

```
void * p_thread_increment (void *arg) {
    int i, val;
    for (i=0; i< ITER ; i++) {
        lock mutex
        if (x == BUFFER_SIZE)
          wait buffer_has_space;

        Access CRITICAL AREA

        signal buffer_has_data
        unlock mutex
    }
    end thread
}
```

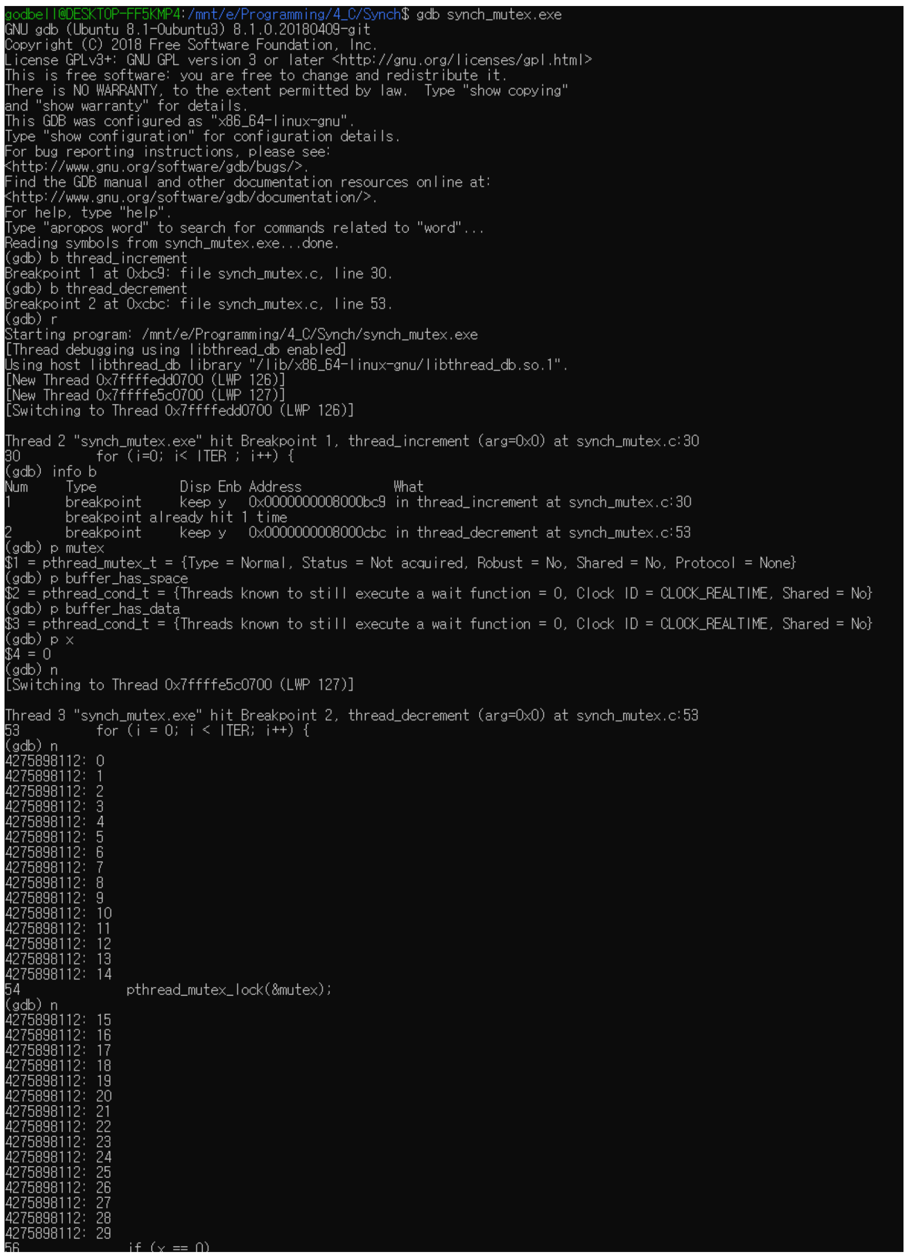
Consumer Thread

```
void * p_thread_decrement (void *arg) {
    int i, val;
    for (i = 0; i < ITER; i++) {
       lock mutex
        if (x == 0)
          wait buffer_has_data;

        Access CRITICAL AREA

        signal buffer_has_space
        unlock mutex
    }
    end thread
}
```

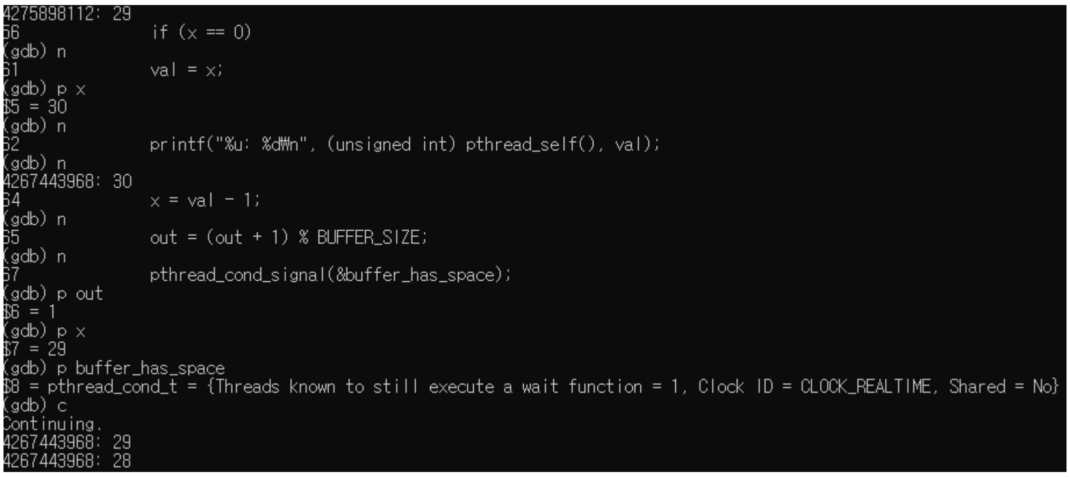
GDB



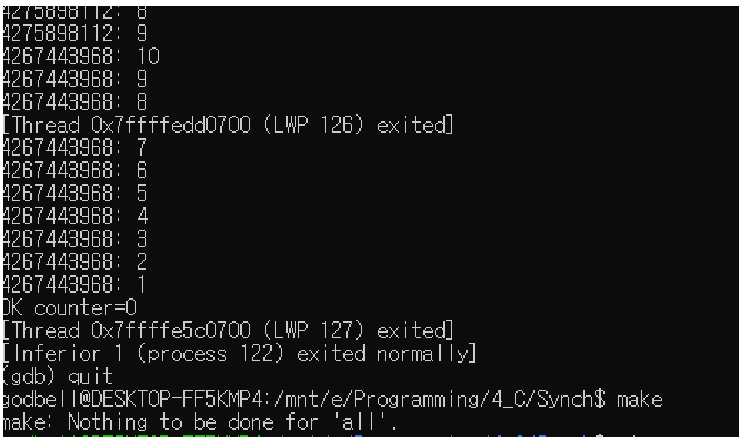
```
godbell@DESKTOP-FF5KMP4:/mnt/e/Programming/4_C/Synch$ gdb synch_mutex.exe
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from synch_mutex.exe...done.
(gdb) b thread_increment
Breakpoint 1 at 0xbc9: file synch_mutex.c, line 30.
(gdb) b thread_decrement
Breakpoint 2 at 0xcbc: file synch_mutex.c, line 53.
(gdb) r
Starting program: /mnt/e/Programming/4_C/Synch/synch_mutex.exe
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7ffffedd0700 (LWP 126)]
[New Thread 0x7ffffe5c0700 (LWP 127)]
[Switching to Thread 0x7ffffedd0700 (LWP 126)]

Thread 2 "synch_mutex.exe" hit Breakpoint 1, thread_increment (arg=0x0) at synch_mutex.c:30
30          for (i=0; i< ITER ; i++) {
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000008000bc9 in thread_increment at synch_mutex.c:30
        breakpoint already hit 1 time
2       breakpoint     keep y   0x0000000008000cbc in thread_decrement at synch_mutex.c:53
(gdb) p mutex
$1 = pthread_mutex_t = {Type = Normal, Status = Not acquired, Robust = No, Shared = No, Protocol = None}
(gdb) p buffer_has_space
$2 = pthread_cond_t = {Threads known to still execute a wait function = 0, Clock ID = CLOCK_REALTIME, Shared = No}
(gdb) p buffer_has_data
$3 = pthread_cond_t = {Threads known to still execute a wait function = 0, Clock ID = CLOCK_REALTIME, Shared = No}
(gdb) p x
$4 = 0
(gdb) n
[Switching to Thread 0x7ffffe5c0700 (LWP 127)]

Thread 3 "synch_mutex.exe" hit Breakpoint 2, thread_decrement (arg=0x0) at synch_mutex.c:53
53          for (i = 0; i < ITER; i++) {
(gdb) n
4275898112: 0
4275898112: 1
4275898112: 2
4275898112: 3
4275898112: 4
4275898112: 5
4275898112: 6
4275898112: 7
4275898112: 8
4275898112: 9
4275898112: 10
4275898112: 11
4275898112: 12
4275898112: 13
4275898112: 14
54                  pthread_mutex_lock(&mutex);
(gdb) n
4275898112: 15
4275898112: 16
4275898112: 17
4275898112: 18
4275898112: 19
4275898112: 20
4275898112: 21
4275898112: 22
4275898112: 23
4275898112: 24
4275898112: 25
4275898112: 26
4275898112: 27
4275898112: 28
4275898112: 29
56                  if (x == 0)
```



```
4275898112: 29
56                  if (x == 0)
(gdb) n
61                  val = x;
(gdb) p x
$5 = 30
(gdb) n
62                  printf("%u: %d₩n", (unsigned int) pthread_self(), val);
(gdb) n
4267443968: 30
64                  x = val - 1;
(gdb) n
65                  out = (out + 1) % BUFFER_SIZE;
(gdb) n
67                  pthread_cond_signal(&buffer_has_space);
(gdb) p out
$6 = 1
(gdb) p x
$7 = 29
(gdb) p buffer_has_space
$8 = pthread_cond_t = {Threads known to still execute a wait function = 1, Clock ID = CLOCK_REALTIME, Shared = No}
(gdb) c
Continuing.
4267443968: 29
4267443968: 28
```



```
4275898112: 8
4275898112: 9
4267443968: 10
4267443968: 9
4267443968: 8
[Thread 0x7ffffedd0700 (LWP 126) exited]
4267443968: 7
4267443968: 6
4267443968: 5
4267443968: 4
4267443968: 3
4267443968: 2
4267443968: 1
OK counter=0
[Thread 0x7ffffe5c0700 (LWP 127) exited]
[Inferior 1 (process 122) exited normally]
(gdb) quit
godbell@DESKTOP-FF5KMP4:/mnt/e/Programming/4_C/Synch$ make
make: Nothing to be done for 'all'.
```

Semaphore를 이용한 Solution Pseudo Code

Variables

```
#define BUFFER_SIZE 30

int x = 0; // count, max 29
int in = 0;
int out = 0;
int buffer[BUFFER_SIZE];

sem_t full; // 30으로 초기화 - 소비자 스레드에서 감소, 생산자 스레드에서 증가
sem_t empty; // 0으로 초기화 - full과 반대로 작용
sem_t mutex; // 1로 초기화 - critical area 접근 시 및 종료 시에 각각 증가 및 감소
```

Producer Thread

```
void * thread_increment (void *arg) {
    int i, val;
    for (i=0; i< ITER ; i++) {
        sem_wait(&empty);
        sem_wait(&mutex);

        Access CRITICAL AREA

        sem_post(&mutex);
        sem_post(&full);
    }
    end thread
}
```

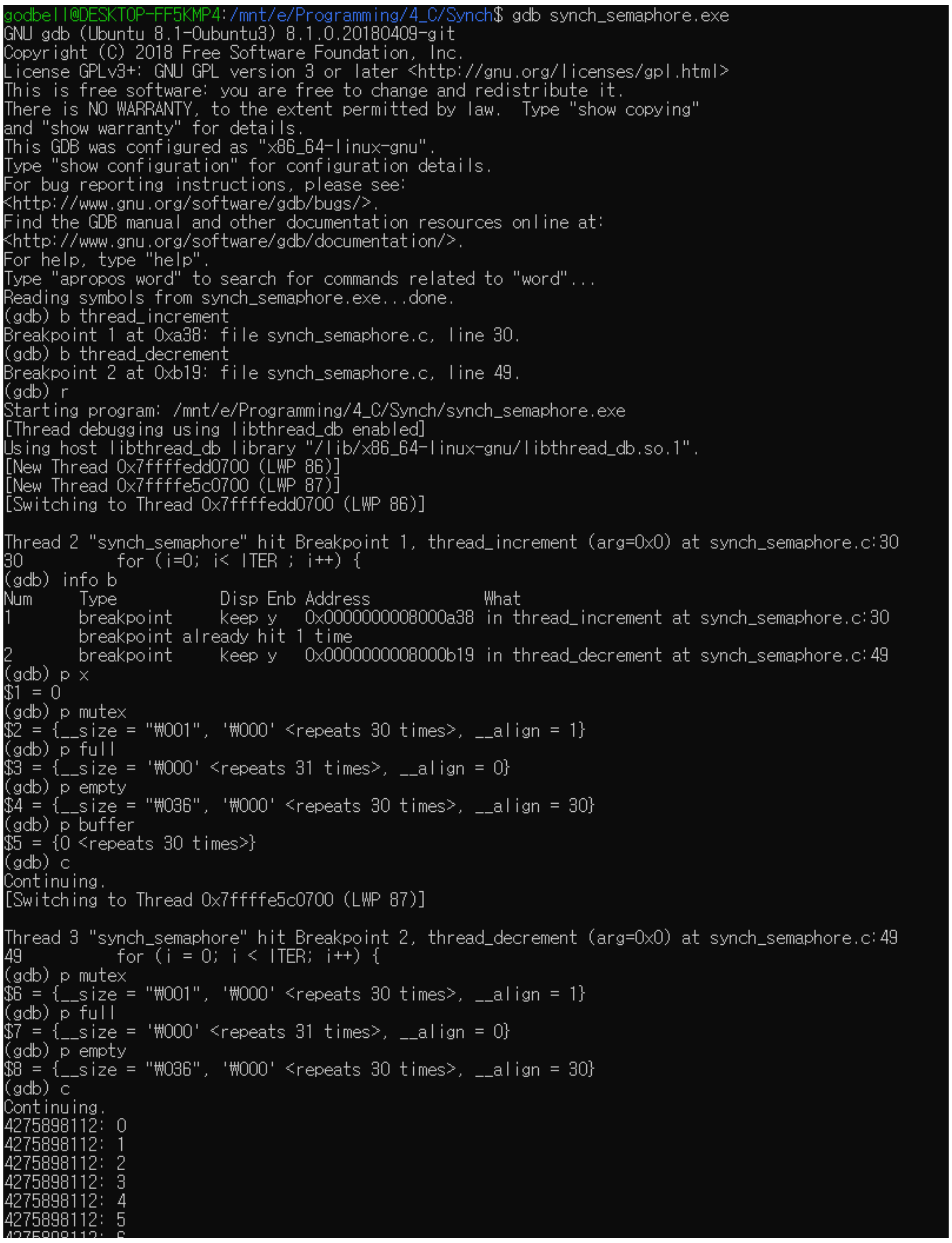
Consumer Thread

```
void * thread_decrement (void *arg) {
    int i, val;
    for (i = 0; i < ITER; i++) {
        sem_wait(&full);
        sem_wait(&mutex);

        Access CRITICAL AREA

        sem_post(&mutex);
        sem_post(&empty);
    }
    end thread
}
```

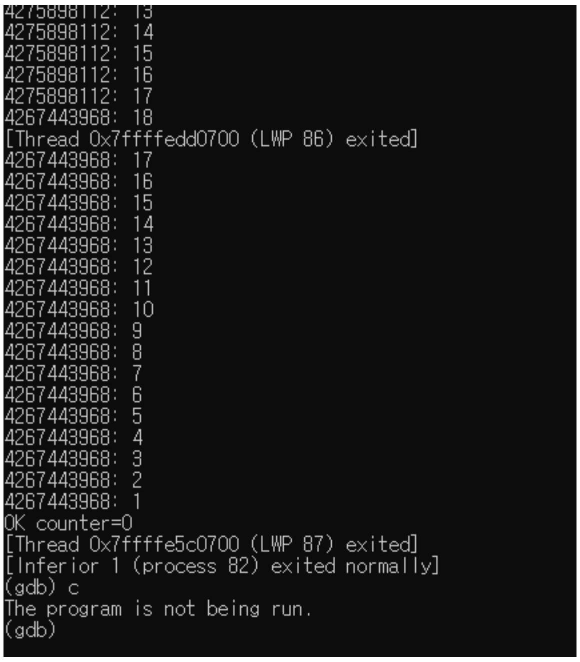
GDB



```
godbell@DESKTOP-FF5KMP4:/mnt/e/Programming/4_C/Synch$ gdb synch_semaphore.exe
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from synch_semaphore.exe...done.
(gdb) b thread_increment
Breakpoint 1 at 0xa38: file synch_semaphore.c, line 30.
(gdb) b thread_decrement
Breakpoint 2 at 0xb19: file synch_semaphore.c, line 49.
(gdb) r
Starting program: /mnt/e/Programming/4_C/Synch/synch_semaphore.exe
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7ffffedd0700 (LWP 86)]
[New Thread 0x7ffffe5c0700 (LWP 87)]
[Switching to Thread 0x7ffffedd0700 (LWP 86)]

Thread 2 "synch_semaphore" hit Breakpoint 1, thread_increment (arg=0x0) at synch_semaphore.c:30
30          for (i=0; i< ITER ; i++) {
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000008000a38 in thread_increment at synch_semaphore.c:30
        breakpoint already hit 1 time
2       breakpoint     keep y   0x0000000008000b19 in thread_decrement at synch_semaphore.c:49
(gdb) p x
$1 = 0
(gdb) p mutex
$2 = {__size = "₩001", '₩000' <repeats 30 times>, __align = 1}
(gdb) p full
$3 = {__size = '₩000' <repeats 31 times>, __align = 0}
(gdb) p empty
$4 = {__size = "₩036", '₩000' <repeats 30 times>, __align = 30}
(gdb) p buffer
$5 = {0 <repeats 30 times>}
(gdb) c
Continuing.
[Switching to Thread 0x7ffffe5c0700 (LWP 87)]

Thread 3 "synch_semaphore" hit Breakpoint 2, thread_decrement (arg=0x0) at synch_semaphore.c:49
49          for (i = 0; i < ITER; i++) {
(gdb) p mutex
$6 = {__size = "₩001", '₩000' <repeats 30 times>, __align = 1}
(gdb) p full
$7 = {__size = '₩000' <repeats 31 times>, __align = 0}
(gdb) p empty
$8 = {__size = "₩036", '₩000' <repeats 30 times>, __align = 30}
(gdb) c
Continuing.
4275898112: 0
4275898112: 1
4275898112: 2
4275898112: 3
4275898112: 4
4275898112: 5
```



```
4275898112: 13
4275898112: 14
4275898112: 15
4275898112: 16
4275898112: 17
4267443968: 18
[Thread 0x7ffffedd0700 (LWP 86) exited]
4267443968: 17
4267443968: 16
4267443968: 15
4267443968: 14
4267443968: 13
4267443968: 12
4267443968: 11
4267443968: 10
4267443968: 9
4267443968: 8
4267443968: 7
4267443968: 6
4267443968: 5
4267443968: 4
4267443968: 3
4267443968: 2
4267443968: 1
OK counter=0
[Thread 0x7ffffe5c0700 (LWP 87) exited]
[Inferior 1 (process 82) exited normally]
(gdb) c
The program is not being run.
(gdb)
```