Reader-Writer Locks: no Starvation

아래 화일에 구현된 Reader-Writer Locks를 기반으로 아래 작업을 수행하시오.
HW-reader-writer.c

가. 쓰레드별 상태 추적 출력하기
- 파라미터 입력 방법 구현 (예: command line parameter 처리)
- 다양한 시나리오 생성 및 설명
- 각 시나리오에 대한 실제 실행 결과

나. Reader-Writer Lock 개선
- Starvation 현상이 제거되도록 개선(Writer가 기다리는데 Reader가 계속 도착하면 발생)
- 힌트:
아래 4개의 카운터를 증감하면서 그 상황에 맞게 기다리거나(Sem_wait) 대기 중인 쓰레드를 깨운다(Sem_post)

AR: number of Active Readers
AW: number of Active Writers
WR: number of Waiting Readers
WW: number of Waiting Writers

- 개선된 Reader-Writer Lock을 사용해서 '가' 에서 사용한 여러 시나리오를 실행하고 결과를 비교

[시나리오 예]

도착 시간(arrival_delay)	Job(thread_id)	종류(job_type)	실행 시간(running_time)
0	0	reader(0)	5
1	1	reader(0)	5
3	2	writer(1)	4
5	3	reader(0)	3
6	4	writer(1)	2
7	5	reader(0)	4

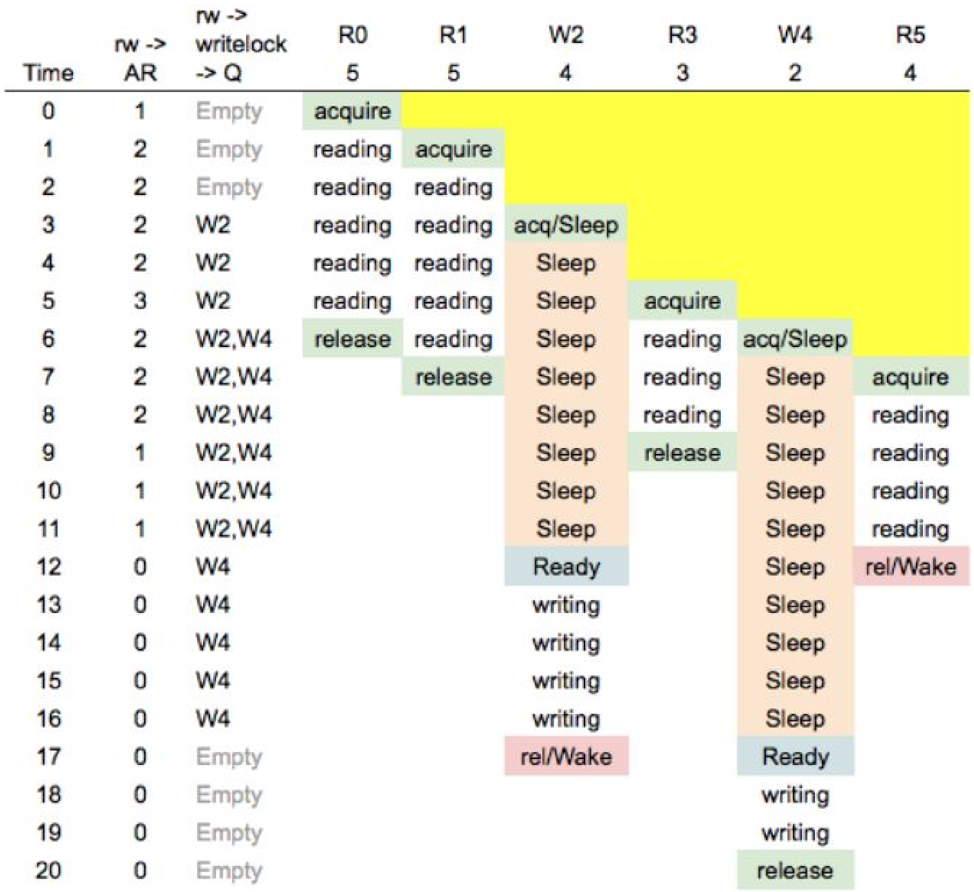
Command line argument 일 경우

```
./reader-writer  -n 6 -a 0:0:5,0:1:8,1:3:4,0:5:7,1:6:2,0:7:4
```

또는

```
./reader-writer  -n 6 -a r:0:5,r:1:8,w:3:4,r:5:7,w:6:2,r:7:4
```

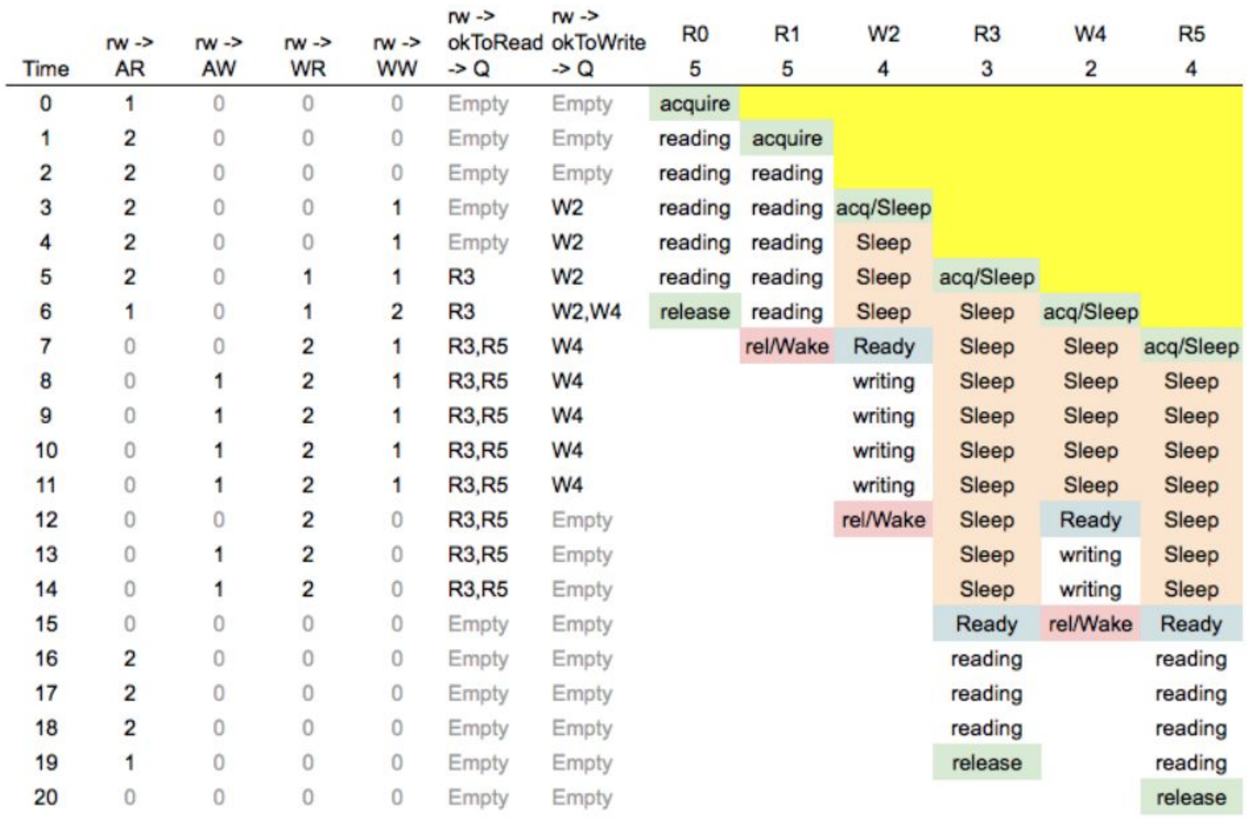
[예제 시나리오의 Thread Trace]



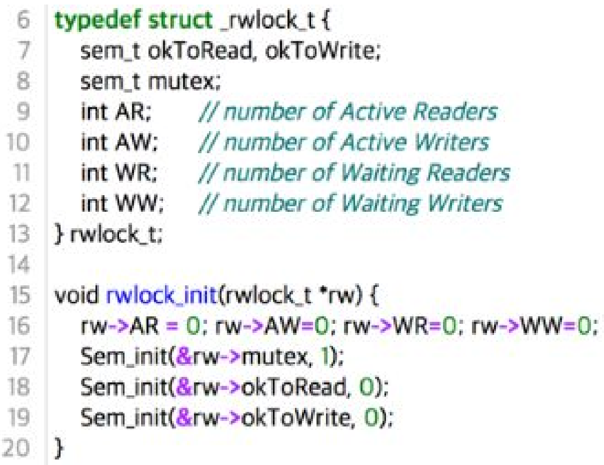
Time	rw -> AR	rw -> writelock -> Q	R0 5	R1 5	W2 4	R3 3	W4 2	R5 4
0	1	Empty	acquire					
1	2	Empty	reading	acquire				
2	2	Empty	reading	reading				
3	2	W2	reading	reading	acq/Sleep			
4	2	W2	reading	reading	Sleep			
5	3	W2	reading	reading	Sleep	acquire		
6	2	W2,W4	release	reading	Sleep	reading	acq/Sleep	
7	2	W2,W4		release	Sleep	reading	Sleep	acquire
8	2	W2,W4			Sleep	reading	Sleep	reading
9	1	W2,W4			Sleep	release	Sleep	reading
10	1	W2,W4			Sleep		Sleep	reading
11	1	W2,W4			Sleep		Sleep	reading
12	0	W4			Ready		Sleep	rel/Wake
13	0	W4			writing		Sleep	
14	0	W4			writing		Sleep	
15	0	W4			writing		Sleep	
16	0	W4			writing		Sleep	
17	0	Empty			rel/Wake		Ready	
18	0	Empty					writing	
19	0	Empty					writing	
20	0	Empty					release	

Thread Trace - RW lock

[Read-Writer Lock이 개선되었을 때의 Thread Trace]

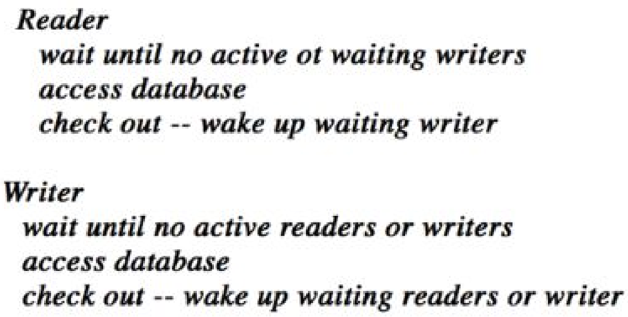


Time	rw -> AR	rw -> AW	rw -> WR	rw -> WW	rw -> okToRead -> Q	rw -> okToWrite -> Q	R0 5	R1 5	W2 4	R3 3	W4 2	R5 4
0	1	0	0	0	Empty	Empty	acquire					
1	2	0	0	0	Empty	Empty	reading	acquire				
2	2	0	0	0	Empty	Empty	reading	reading				
3	2	0	0	1	Empty	W2	reading	reading	acq/Sleep			
4	2	0	0	1	Empty	W2	reading	reading	Sleep			
5	2	0	1	1	R3	W2	reading	reading	Sleep	acq/Sleep		
6	1	0	1	2	R3	W2,W4	release	reading	Sleep	Sleep	acq/Sleep	
7	0	0	2	1	R3,R5	W4		rel/Wake	Ready	Sleep	Sleep	acq/Sleep
8	0	1	2	1	R3,R5	W4			writing	Sleep	Sleep	Sleep
9	0	1	2	1	R3,R5	W4			writing	Sleep	Sleep	Sleep
10	0	1	2	1	R3,R5	W4			writing	Sleep	Sleep	Sleep
11	0	1	2	1	R3,R5	W4			writing	Sleep	Sleep	Sleep
12	0	0	2	0	R3,R5	Empty			rel/Wake	Sleep	Ready	Sleep
13	0	1	2	0	R3,R5	Empty				Sleep	writing	Sleep
14	0	1	2	0	R3,R5	Empty				Sleep	writing	Sleep
15	0	0	0	0	Empty	Empty				Ready	rel/Wake	Ready
16	2	0	0	0	Empty	Empty				reading		reading
17	2	0	0	0	Empty	Empty				reading		reading
18	2	0	0	0	Empty	Empty				reading		reading
19	1	0	0	0	Empty	Empty				release		reading
20	0	0	0	0	Empty	Empty						release



```
typedef struct _rwlock_t {
    sem_t okToRead, okToWrite;
    sem_t mutex;
    int AR;     // number of Active Readers
    int AW;     // number of Active Writers
    int WR;     // number of Waiting Readers
    int WW;     // number of Waiting Writers
} rwlock_t;

void rwlock_init(rwlock_t *rw) {
    rw->AR = 0; rw->AW=0; rw->WR=0; rw->WW=0;
    Sem_init(&rw->mutex, 1);
    Sem_init(&rw->okToRead, 0);
    Sem_init(&rw->okToWrite, 0);
}
```

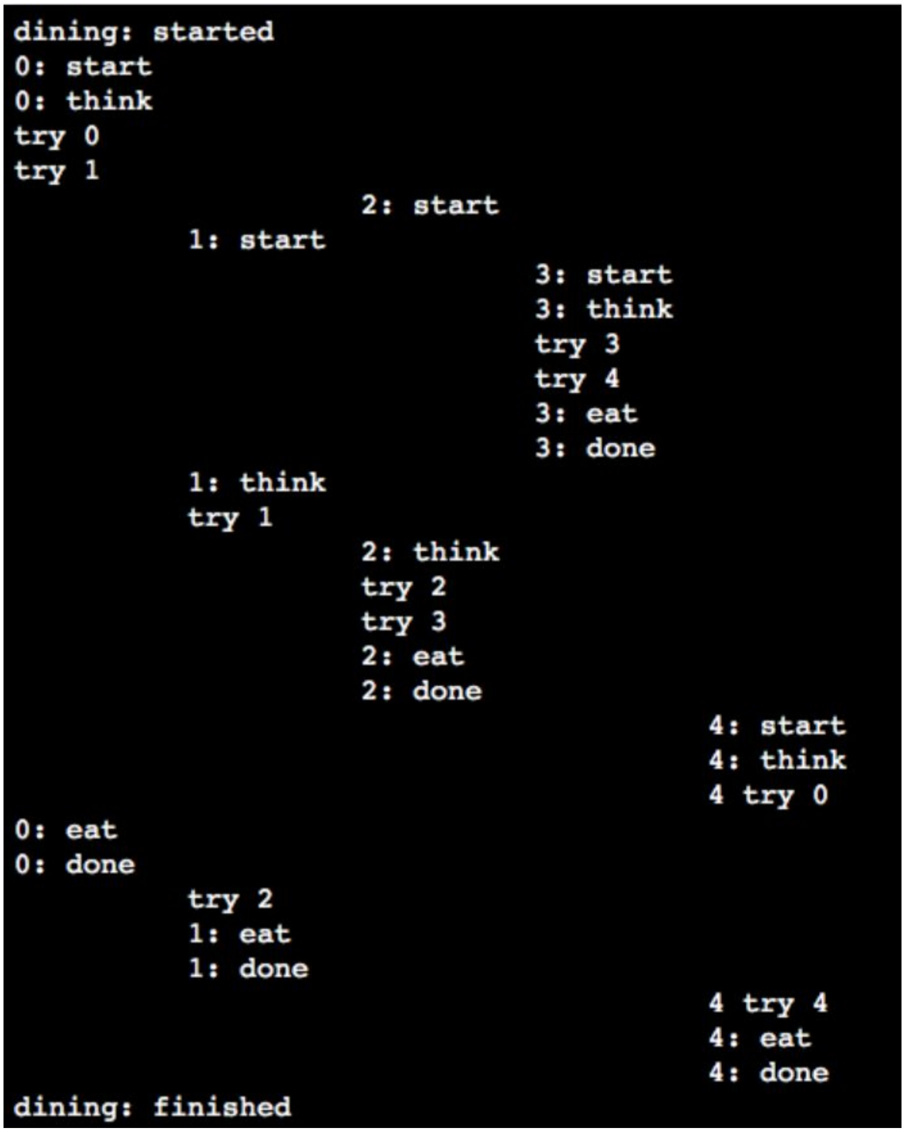


```
Reader
  wait until no active ot waiting writers
  access database
  check out -- wake up waiting writer

Writer
  wait until no active readers or writers
  access database
  check out -- wake up waiting readers or writer
```

Thread Trace - RW lock

[참고]
각 쓰레드 상태를 추적하며 열을 맞춰서 출력하는 예제
ostep-code/threads-sema/dining_philosophers_no_deadlock_print.c



```
dining: started
0: start
0: think
try 0
try 1
                    2: start
          1: start
                              3: start
                              3: think
                              try 3
                              try 4
                              3: eat
                              3: done
          1: think
          try 1
                    2: think
                    try 2
                    try 3
                    2: eat
                    2: done
                                        4: start
                                        4: think
                                        4 try 0
0: eat
0: done
          try 2
          1: eat
          1: done
                                        4 try 4
                                        4: eat
                                        4: done
dining: finished
```