

운영체제론 Project 3

2017012251 윤영훈

소스코드 설명 - writer_prefer.c

```
254 pthread_mutex_t mutex;  
255 pthread_cond_t rcond;  
256 pthread_cond_t wcond;  
257  
258 int writer_count = 0;  
259 int writer_act = 0;  
260  
261 int reader_act = 0;
```

전역변수 선언

교수님의 reader_prefer를 참고, Condition Variable을 사용하여 writer_prefer를 구현.

line 254를 통해 Reader와 Writer의 상호배타용 mutex를 선언.

line 255, 256을 통해 각각 Reader와 Writer의 Condition Variable인 rcond, wcond를 선언.

또한 Writer가 중복되는 경우를 막기 위해 writer_act를, Reader가 끝나면 Writer에게 순서를 넘겨주기 위해 reader_act를, 더 이상 대기중인 Writer가 없을 때 Reader에게 순서를 넘겨주기 위해 writer_count를 선언.

```
263 void *reader(void *arg)  
264 {  
265     int id, i;  
266  
267     /*  
268      * 들어온 인자를 통해 출력할 문자의 종류를 정한다.  
269      */  
270     id = *(int *)arg;  
271     /*  
272      * 스레드가 살아 있는 동안 같은 문자열 시퀀스 <XXX...XX>를 반복해서 출력한다.  
273      */  
274  
275     while (alive) {  
276  
277         pthread_mutex_lock(&mutex);  
278  
279         while (writer_count > 0 || writer_act == 1){  
280             pthread_cond_wait(&rcond, &mutex);  
281         }  
282         reader_act++;  
283  
284         pthread_mutex_unlock(&mutex);  
285  
286         /*  
287          * Begin Critical Section  
288          */  
289         printf("<");  
290         for (i = 0; i < N; ++i)  
291             printf("%c", 'A'+id);  
292         printf(">");  
293         /*  
294          * End Critical Section  
295          */  
296  
297         pthread_mutex_lock(&mutex);  
298  
299         reader_act--;  
300  
301         if(reader_act == 0)  
302             pthread_cond_signal(&wcond);  
303  
304         pthread_mutex_unlock(&mutex);  
305     }  
306     pthread_exit(0);  
307 }  
308 }
```

Reader는 line 277을 통해 mutex lock을 시작, 이때 writer_count > 0 또는 writer_act == 1이라면 wait.

해당 조건을 만족하여 while문을 벗어난다면 CS에 진입할 준비가 된 것이므로 reader_act를 증가시키고 mutex lock을 해제, CS에 진입한다.

CS를 빠져나오면 다시 mutex lock을 걸어주고, reader_act를 감소시키고 line 302를 통해 wcond를 깨운 뒤 mutex lock을 해제 후 종료.

Reader

```

316 void *writer(void *arg)
317 {
318     int id, i;
319     struct timespec req, rem;
320
321     /*
322      * 들어온 인자를 통해 얼굴 이미지의 종류를 정한다.
323      */
324     id = *(int *)arg;
325     /*
326      * 이미지 출력을 천천히 하기 위해 한 줄 출력할 때마다 쉬는 시간을 1 나노초로 설정한다.
327      */
328     req.tv_sec = 0;
329     req.tv_nsec = 1;
330     /*
331      * 스레드가 살아 있는 동안 같은 이미지를 반복해서 출력한다.
332      */
333     while (alive) {
334         pthread_mutex_lock(&mutex);
335
336         writer_count++;
337         pthread_cond_wait(&wcond, &mutex);
338
339         writer_act++;
340
341         pthread_mutex_unlock(&mutex);
342
343         /*
344          * Begin Critical Section
345          */

```

Writer 1

Writer는 line 335를 통해 mutex lock에 진입하고 이때 writer_count를 증가시키면서 line 338에서 대기 후 CS에 진입할 준비가 되면 writer_act를 증가시키면서 mutex lock을 해제 후 CS에 진입.

```

370     /*
371      * End Critical Section
372      */
373
374     pthread_mutex_lock(&mutex);
375
376     writer_count--;
377     writer_act--;
378
379     if(writer_count == 0)
380         pthread_cond_broadcast(&rcond);
381     else
382         pthread_cond_signal(&wcond);
383
384     pthread_mutex_unlock(&mutex);
385
386 }
387 pthread_exit(0);
388 }

```

CS를 빠져나오면 다시 mutex lock을 걸고 writer_count와 writer_act를 감소시킨다.

이때 Writer Prefer이므로, 더 이상 대기중인 Writer가 없다면 broadcast를 통해 rcond에게 알려주어 순서를 넘겨주고, 대기중인 Writer가 있다면 다음 Writer에게 순서를 넘겨준다.

그 후 mutex lock을 해제하고 종료한다.

Writer 2

```

395 int main(void)
396 {
397     int i;
398     int rarg[RNUM], warg[WNUM];
399     pthread_t rthid[RNUM];
400     pthread_t wthid[WNUM];
401     struct timespec req, rem;
402
403     pthread_mutex_init(&mutex, NULL);
404     pthread_cond_init(&rcond, NULL);
405     pthread_cond_init(&wcond, NULL);
406
407     /*
408     * Create RNUM reader threads
409     */
410     for (i = 0; i < RNUM; ++i) {
411         rarg[i] = i;
412         if (pthread_create(rthid+i, NULL, reader, rarg+i) != 0) {
413             fprintf(stderr, "pthread_create error\n");
414             exit(-1);
415         }
416     }
417     /*
418     * Create WNUM writer threads
419     */
420     for (i = 0; i < WNUM; ++i) {
421         warg[i] = i;
422         if (pthread_create(wthid+i, NULL, writer, warg+i) != 0) {
423             fprintf(stderr, "pthread_create error\n");
424             exit(-1);
425         }
426     }
427     /*
428     * Wait for 5.0 second while the threads are working
429     */
430     req.tv_sec = 5;
431     req.tv_nsec = 000000000L;
432     nanosleep(&req, &rem);

```

Main

line 403 - 405를 통해 선언해둔 mutex, rcond, wcond를 초기화하고, 2MB 내외의 출력물을 확보하기 위해 line 430 - 431에서 대기 시간을 5초로 늘린다. ➔ 출력물의 크기 : 약 1.7MB

소스코드 설명 - fair_reader_writer.c

```
254 pthread_mutex_t rw_mutex;
255 pthread_mutex_t fair_mutex;
256 pthread_mutex_t reader_count_mutex;
257
258 int read_count = 0;
```

전역변수 선언

‘다만 조건변수를 사용하여 공정한 reader_writer 방식을 구현하는 것은 어려울 수 있다’는 교수님의 프로젝트 조언에 따라 Condition Variable을 포기하고 대신 강의 자료 ch7-annotated의 7 - 10페이지를 참고하여 Reader와 Writer의 상호배타용 rw_mutex를, read_count를 업데이트하기 위해 reader_count_mutex를, 진행중인 reader의 수를 파악하기 위해 read_count를 선언한다. 이때 fair 조건을 만족하기 위해 line 255의 fair_mutex를 추가로 선언한다.

```
260 void *reader(void *arg)
261 {
262     int id, i;
263
264     /*
265      * 들어온 인자를 통해 출력할 문자의 종류를 정한다.
266      */
267     id = *(int *)arg;
268     /*
269      * 스레드가 살아 있는 동안 같은 문자열 시퀀스 <xxx...xx>를 반복해서 출력한다.
270      */
271
272     while (alive) {
273
274         pthread_mutex_lock(&fair_mutex);
275         pthread_mutex_lock(&reader_count_mutex);
276
277         read_count++;
278         if(read_count == 1)
279             pthread_mutex_lock(&rw_mutex);
280
281         pthread_mutex_unlock(&reader_count_mutex);
282         pthread_mutex_unlock(&fair_mutex);
283
284         /*
285          * Begin Critical Section
286          */
287         printf("<");
288         for (i = 0; i < N; ++i)
289             printf("%c", 'A'+id);
290         printf(">");
291         /*
292          * End Critical Section
293          */
294
295         pthread_mutex_lock(&reader_count_mutex);
296
297         read_count--;
298         if(read_count == 0)
299             pthread_mutex_unlock(&rw_mutex);
300
301         pthread_mutex_unlock(&reader_count_mutex);
302     }
303     pthread_exit(0);
304 }
```

fair한 순서를 위해 우선 line 274의 fair_mutex lock을 먼저 걸어주고, 이후 read_count 증가를 위해 reader_count_mutex lock을 걸어준다.

이후 read_count를 증가시키고, 만약 자신이 유일한 Reader라면 Writer와 경쟁하고 그렇지 않다면 다른 Reader가 경쟁하는 것을 허용한다. 그 후 mutex unlock을 처리한 뒤 CS에 진입한다.

CS에서 빠져나오면 read_count를 감소시키며, 더 이상의 Reader가 없다면 Writer에게 순서를 넘겨주며 mutex unlock 후 종료한다.

Reader

```

312 void *writer(void *arg)
313 {
314     int id, i;
315     struct timespec req, rem;
316
317     /*
318      * 들어온 인자를 통해 얼굴 이미지의 종류를 정한다.
319      */
320     id = *(int *)arg;
321     /*
322      * 이미지 출력을 천천히 하기 위해 한 줄 출력할 때마다 쉬는 시간을 1 나노초로 설정한다.
323      */
324     req.tv_sec = 0;
325     req.tv_nsec = 1L;
326     /*
327      * 스레드가 살아 있는 동안 같은 이미지를 반복해서 출력한다.
328      */
329     while (alive) {
330
331         pthread_mutex_lock(&fair_mutex);
332         pthread_mutex_lock(&rw_mutex);
333         pthread_mutex_unlock(&fair_mutex);
334
335         /*
336          * Begin Critical Section
337          */

```

Writer 1

Writer도 마찬가지로 fair한 순서를 위해 fair_mutex lock을 우선 걸어준 뒤 line 332를 통해 Reader와 경쟁한다. 자신의 순서가 되면 fair_mutex lock을 해제한 뒤 CS에 진입한다.

```

361         /*
362          * End Critical Section
363          */
364
365         pthread_mutex_unlock(&rw_mutex);
366
367     }
368     pthread_exit(0);
369 }

```

Writer 2

CS를 빠져나오면 rw_mutex lock을 해제하면서 Reader에게 순서를 넘겨준다. fair한 순서를 위해 fair mutex를 따로 선언했고, 자세한 조건 처리 과정들은 Reader에서 처리하였으므로 상대적으로 Writer의 코드는 간단하다.

```

376 int main(void)
377 {
378     int i;
379     int rang[RNUM], warg[WNUM];
380     pthread_t rthid[RNUM];
381     pthread_t wthid[WNUM];
382     struct timespec req, rem;
383
384     pthread_mutex_init(&rw_mutex, NULL);
385     pthread_mutex_init(&fair_mutex, NULL);
386     pthread_mutex_init(&reader_count_mutex, NULL);
387
388     /*
389      * Create RNUM reader threads
390      */
391     for (i = 0; i < RNUM; ++i) {
392         rang[i] = i;
393         if (pthread_create(rthid+i, NULL, reader, rang+i) != 0) {
394             fprintf(stderr, "pthread_create error\n");
395             exit(-1);
396         }
397     }
398     /*
399      * Create WNUM writer threads
400      */
401     for (i = 0; i < WNUM; ++i) {
402         warg[i] = i;
403         if (pthread_create(wthid+i, NULL, writer, warg+i) != 0) {
404             fprintf(stderr, "pthread_create error\n");
405             exit(-1);
406         }
407     }
408     /*
409      * Wait for 2.0 second while the threads are working
410      */
411     req.tv_sec = 2;
412     req.tv_nsec = 000000000L;
413     nanosleep(&req, &rem);

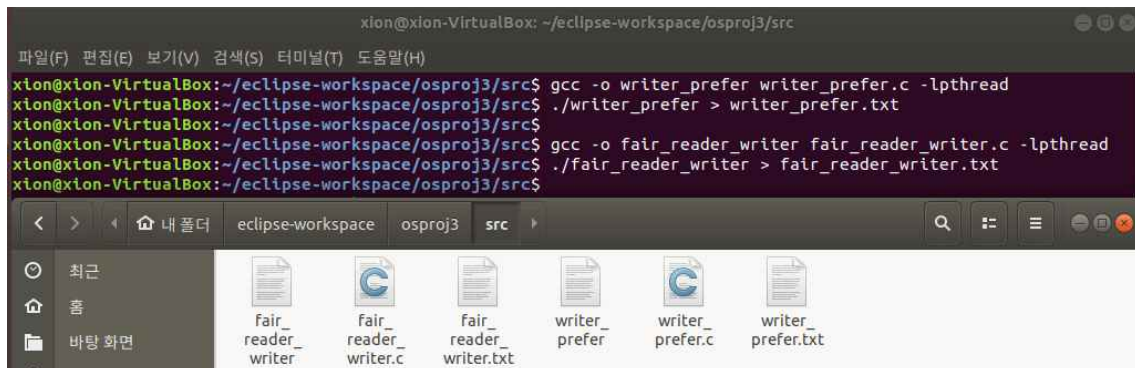
```

Main

line 384 - 386을 통해 선언해둔 rw_mutex, fair_mutex, reader_count_mutex를 초기화하고,
2MB 내외의 출력물을 확보하기 위해 line 411 - 412를 통해 대기 시간을 2초로 늘린다.

➔ 출력물의 크기 : 약 2.4MB

컴파일 과정



```
xion@xion-VirtualBox: ~/eclipse-workspace/osproj3/src
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
xion@xion-VirtualBox:~/eclipse-workspace/osproj3/src$ gcc -o writer_prefer writer_prefer.c -lpthread
xion@xion-VirtualBox:~/eclipse-workspace/osproj3/src$ ./writer_prefer > writer_prefer.txt
xion@xion-VirtualBox:~/eclipse-workspace/osproj3/src$ gcc -o fair_reader_writer fair_reader_writer.c -lpthread
xion@xion-VirtualBox:~/eclipse-workspace/osproj3/src$ ./fair_reader_writer > fair_reader_writer.txt
xion@xion-VirtualBox:~/eclipse-workspace/osproj3/src$
```

출력물 점검

교수님께서 공지로 올려주신 ‘프로젝트 #3 오동작 간단한 판별법’에 의거,

writer_prefer의 경우

1. 마지막 Reader가 종료된 후 Writer의 인물 사진이 지루하도록 연속적으로 나오는 결과를 확인
2. Writer가 끝난 뒤 마지막으로 Reader들이 중복해서 나오는 결과 확인
3. 테레사 수녀, 로버트 드니로, 아인슈타인의 얼굴 모두가 한 번 이상씩 출력됨을 확인

세 가지 조건을 모두 만족함을 확인.

fair_reader_writer의 경우

Reader와 Writer가 굼주리지 않고 어느 정도 fair한 순서대로 출력되는 결과를 확인.