Operating System Project #3

2017012351 이영섭

내용

알고리즘

writer 선호 알고리즘

reader start

1. mutex_lock을 이용해 잠근다.
2. writer_wating에 대기중인 쓰레드가 있거나, writer가 실행중이라면 조건 변수 cond_r을 wait시킨다.
3. cond이 signal을 받아 다시 실행되면 reader가 실행되고 있는 개수를 나타내는 변수를 +1해준다.
4. mutex_unlock을 이용해 잠금을 푼다.

reader end

1. mutex_lock을 이용해 잠근다.
2. reader가 실행되고 있는 개수를 나타내는 변수를 -1해준다.
3. 만약 실행되고 있는 reader의 개수가 0이라면 조건 변수 cond에서 하나의 쓰레드를 깨운다.(무조건 writer)
4. mutex_unlock을 이용해 잠금을 푼다.

writer start

1. mutex_lock을 이용해 잠근다.
2. 대기하고 있는 writer의 수를 나타내는 변수를 +1해준다.
3. 실행되고 있는 reader의 수가 0보다 크거나 writer가 현재 실행중인동안 조건 변수 cond을 wait시킨다.
4. signal을 받아 쓰레드가 실행되면 대기하고 있는 writer의 수를 나타내는 변수를 -1해준다.
5. writer가 현재 작동중인 것을 나타내는 변수를 1로 설정한다.
6. mutex_unlock을 이용해 잠금을 푼다.

writer end

1. mutex_lock을 이용해 잠근다.
2. writer가 현재 작동중인 것을 나타내는 변수를 0로 설정한다.
3. writer가 기다리는 개수가 0보다 크면 cond 중 하나를 깨우고 아니라면 cond_r에서 reader들을 전부 깨운다.
4. mutex_unlock을 이용해 잠금을 푼다.

fair reader writer 알고리즘

reader start

1. 첫번째 mutex_lock을 이용해 잠근다.(들어오는 순서를 정해주는 mutex_lock)
2. 두번째 mutex_lock을 이용해 reader의 수를 조절한다.(reader_lock)
3. 만약 실행중인 reader가 없다면 reader를 위해 reader resource에 대한 독점 액세스를 요청하고(access_lock)
4. 액세스를 획득하면 reader의 값을 하나 증가시켜 준다.
5. mutex_lock과 reader_lock의 잠금을 푼다.

reader end

1. reader_lock을 잠근다.
2. reader가 하나 종료되어 lock을 획득하였기 때문에 reader값을 하나 줄인다.
3. 만약 실행중인 reader가 없다면 access_lock의 잠금을 풀고 다른 쓰레드가 접근할 수 있도록 한다.

writer start

1. mutex_lock을 잠근다.(들어오는 순서 정해줌)
2. lock을 얻으면 access_lock을 잠금으로써 write resource에 대한 독점 액세스를 요청한다.
3. mutex_lock의 잠금을 풀어서 다음 스레드가 들어오게 한다.

write end

1. writer에서 할 일을 마치고 나면 access_lock의 잠금을 풀어 다른 쓰레드가 접근할 수 있도록 한다.

프로그램 소스코드

wirter_prefer.c

```c
/*
 * Copyright 2020, 2021. Heekuck Oh, all rights reserved
 * 이 프로그램은 한양대학교 ERICA 소프트웨어학부 재학생을 위한 교육용으로 제작되었습니다.
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>

#define N 8192
#define L1 75
#define L2 70
#define L3 70
#define RNUM 10
#define WNUM 3

char *t[L1] = {};

char *d[L2] = {};

char *e[L3] = {};

/*
 * alive 값이 1이면 각 스레드는 무한 루프를 돌며 반복해서 일을 하고,
 * alive 값이 0이 되면 무한 루프를 빠져나와 스레드를 자연스럽게 종료한다.
 */
int alive = 1;
pthread_mutex_t mutex_lock;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
pthread_cond_t cond_r = PTHREAD_COND_INITIALIZER;
int num_readers_active, num_writers_waiting = 0;
int writer_active = 0;

/*
 * Reader 스레드는 같은 문자를 N번 출력한다. 예를 들면 <AAA...AA> 이런 식이다.
 * 출력할 문자는 인자를 통해 0이면 A, 1이면 B, ..., 등으로 출력하며, 시작과 끝을 <...>로 나타낸다.
 * 단일 reader라면 <AAA...AA>처럼 같은 문자만 출력하겠지만, critical section에서 reader의
 * 중복을 허용하기 때문에 reader가 많아지면 출력이 어지럽게 섞여서 나오는 것이 정상이다.
 */

void *reader(void *arg)
```

```
{
    int id, i;

    /*
     * 들어온 인자를 통해 출력할 문자의 종류를 정한다.
     */
    id = *(int *)arg;
    /*
     * 스레드가 살아 있는 동안 같은 문자열 시퀀스 <XXX...XX>를 반복해서 출력한다.
     */
    while (alive) {

        pthread_mutex_lock(&mutex_lock);
        while(num_writers_waiting > 0 || writer_active == 1){
          pthread_cond_wait(&cond_r, &mutex_lock);
        }
        num_readers_active++;
        pthread_mutex_unlock(&mutex_lock);

        /*
         * Begin Critical Section
         */

        printf("<");
        for (i = 0; i < N; ++i)
            printf("%c", 'A'+id);
        printf(">");

        /*
         * End Critical Section
         */

        pthread_mutex_lock(&mutex_lock);
        num_readers_active--;
        if(num_readers_active == 0){
          pthread_cond_signal(&cond);
        }
        pthread_mutex_unlock(&mutex_lock);

    }
    pthread_exit(0);
}

/*
 * Writer 스레드는 어떤 사람의 얼굴 이미지를 출력한다.
 * 이미지는 세 가지 종류가 있으며 인자를 통해 식별한다.
 * Writer 가 critical section 에 있으면 다른 writer 는 물론이고 어떠한 reader 도 들어올 수 없다.
```

```
 * 만일 이것을 어기고 다른 writer나 reader가 들어왔다면 얼굴 이미지가 깨져서 쉽게 감지된다.
 */
void *writer(void *arg)
{
    int id, i;
    struct timespec req, rem;

    /*
     * 들어온 인자를 통해 얼굴 이미지의 종류를 정한다.
     */
    id = *(int *)arg;
    /*
     * 이미지 출력을 천천히 하기 위해 한 줄 출력할 때마다 쉬는 시간을 1 나노초로 설정한다.
     */
    req.tv_sec = 0;
    req.tv_nsec = 1L;
    /*
     * 스레드가 살아 있는 동안 같은 이미지를 반복해서 출력한다.
     */
    while (alive) {

        pthread_mutex_lock(&mutex_lock);
        num_writers_waiting++;
        while(num_readers_active > 0 || writer_active == 1){
          pthread_cond_wait(&cond, &mutex_lock);
        }
        num_writers_waiting--;
        writer_active = 1;
        pthread_mutex_unlock(&mutex_lock);

        /*
         * Begin Critical Section
         */

        printf("\n");
        switch (id) {
            case 0:
                for (i = 0; i < L1; ++i) {
                    printf("%s\n", t[i]);
                    nanosleep(&req, &rem);
                }
                break;
            case 1:
                for (i = 0; i < L2; ++i) {
                    printf("%s\n", d[i]);
                    nanosleep(&req, &rem);
```

```
                }
                break;
            case 2:
                for (i = 0; i < L3; ++i) {
                    printf("%s\n", e[i]);
                    nanosleep(&req, &rem);
                }
                break;
            default:
                ;
        }

        /*
         * End Critical Section
         */

         pthread_mutex_lock(&mutex_lock);
         writer_active = 0;
         if(num_writers_waiting > 0) pthread_cond_signal(&cond);
         else pthread_cond_broadcast(&cond_r);
         pthread_mutex_unlock(&mutex_lock);

    }
    pthread_exit(0);
}

/*
 * 메인 함수는 RNUM 개의 reader 스레드를 생성하고, WNUM 개의 writer 스레드를 생성한다.
 * 생성된 스레드가 일을 할 동안 0.2 초 동안 기다렸다가 alive 의 값을 0 으로 바꿔서 모든 스레드가
 * 무한 루프를 빠져나올 수 있게 만든 후, 스레드가 자연스럽게 종료할 때까지 기다리고 메인을 종료한다.
 */
int main(void)
{
    int i;
    int rarg[RNUM], warg[WNUM];
    pthread_t rthid[RNUM];
    pthread_t wthid[WNUM];
    struct timespec req, rem;
    pthread_mutex_init(&mutex_lock, NULL);
    /*
     * Create RNUM reader threads
     */
    for (i = 0; i < RNUM; ++i) {
        rarg[i] = i;
        if (pthread_create(rthid+i, NULL, reader, rarg+i) != 0) {
```

```
            fprintf(stderr, "pthread_create error\n");
            exit(-1);
        }
    }
    /*
     * Create WNUM writer threads
     */
    for (i = 0; i < WNUM; ++i) {
        warg[i] = i;
        if (pthread_create(wthid+i, NULL, writer, warg+i) != 0) {
            fprintf(stderr, "pthread_create error\n");
            exit(-1);
        }
    }
    /*
     * Wait for 0.2 second while the threads are working
     */
    req.tv_sec = 0;
    req.tv_nsec = 200000000L;
    nanosleep(&req, &rem);
    /*
     * Now terminate all threads and leave
     */
    alive = 0;
    for (i = 0; i < RNUM; ++i)
        pthread_join(rthid[i], NULL);
    for (i = 0; i < WNUM; ++i)
        pthread_join(wthid[i], NULL);
    exit(0);

    pthread_mutex_destroy(&mutex_lock);
}
```

fair_reader_writer.c

```
/*
 * Copyright 2020, 2021. Heekuck Oh, all rights reserved
 * 이 프로그램은 한양대학교 ERICA 소프트웨어학부 재학생을 위한 교육용으로 제작되었습니다.
 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>

#define N 8192
#define L1 75
#define L2 70
```

```c
#define L3 70
#define RNUM 10
#define WNUM 3

char *t[L1] = {};

char *d[L2] = {};

char *e[L3] = {};

/*
 * alive 값이 1 이면 각 스레드는 무한 루프를 돌며 반복해서 일을 하고,
 * alive 값이 0 이 되면 무한 루프를 빠져나와 스레드를 자연스럽게 종료한다.
 */
int alive = 1;
pthread_mutex_t mutex_lock, reader_lock, access_lock;
int readers = 0;

/*
 * Reader 스레드는 같은 문자를 N 번 출력한다. 예를 들면 <AAA...AA> 이런 식이다.
 * 출력할 문자는 인자를 통해 0 이면 A, 1 이면 B, ..., 등으로 출력하며, 시작과 끝을 <
...>로 나타낸다.
 * 단일 reader 라면 <AAA...AA>처럼 같은 문자만 출력하겠지만, critical section 에서
 reader 의
 * 중복을 허용하기 때문에 reader 가 많아지면 출력이 어지럽게 섞여서 나오는 것이 정상
이다.
 */
void *reader(void *arg)
{
    int id, i;

    /*
     * 들어온 인자를 통해 출력할 문자의 종류를 정한다.
     */
    id = *(int *)arg;
    /*
     * 스레드가 살아 있는 동안 같은 문자열 시퀀스 <XXX...XX>를 반복해서 출력한다.
     */
    while (alive) {

        pthread_mutex_lock(&mutex_lock);
        pthread_mutex_lock(&reader_lock);
        if(readers == 0){
          pthread_mutex_lock(&access_lock);
        }
        readers++;
        pthread_mutex_unlock(&mutex_lock);
        pthread_mutex_unlock(&reader_lock);
```

```
        /*
         * Begin Critical Section
         */

        printf("<");
        for (i = 0; i < N; ++i)
            printf("%c", 'A'+id);
        printf(">");

        /*
         * End Critical Section
         */

        pthread_mutex_lock(&reader_lock);
        readers--;
        if(readers == 0){
          pthread_mutex_unlock(&access_lock);
        }
        pthread_mutex_unlock(&reader_lock);

    }
    pthread_exit(0);
}

/*
 * Writer 스레드는 어떤 사람의 얼굴 이미지를 출력한다.
 * 이미지는 세 가지 종류가 있으며 인자를 통해 식별한다.
 * Writer 가 critical section 에 있으면 다른 writer 는 물론이고 어떠한 reader 도 들어올 수 없다.
 * 만일 이것을 어기고 다른 writer 나 reader 가 들어왔다면 얼굴 이미지가 깨져서 쉽게 감지된다.
 */
void *writer(void *arg)
{
    int id, i;
    struct timespec req, rem;

    /*
     * 들어온 인자를 통해 얼굴 이미지의 종류를 정한다.
     */
    id = *(int *)arg;
    /*
     * 이미지 출력을 천천히 하기 위해 한 줄 출력할 때마다 쉬는 시간을 1 나노초로 설정한다.
     */
    req.tv_sec = 0;
    req.tv_nsec = 1L;
```

```
    /*
     * 스레드가 살아 있는 동안 같은 이미지를 반복해서 출력한다.
     */
    while (alive) {

        pthread_mutex_lock(&mutex_lock);
        pthread_mutex_lock(&access_lock);
        pthread_mutex_unlock(&mutex_lock);

        /*
         * Begin Critical Section
         */

        printf("\n");
        switch (id) {
            case 0:
                for (i = 0; i < L1; ++i) {
                    printf("%s\n", t[i]);
                    nanosleep(&req, &rem);
                }
                break;
            case 1:
                for (i = 0; i < L2; ++i) {
                    printf("%s\n", d[i]);
                    nanosleep(&req, &rem);
                }
                break;
            case 2:
                for (i = 0; i < L3; ++i) {
                    printf("%s\n", e[i]);
                    nanosleep(&req, &rem);
                }
                break;
            default:
                ;
        }

        /*
         * End Critical Section
         */

        pthread_mutex_unlock(&access_lock);

    }
    pthread_exit(0);
}

/*
```

```
 * 메인 함수는 RNUM 개의 reader 스레드를 생성하고, WNUM 개의 writer 스레드를 생성
한다.
 * 생성된 스레드가 일을 할 동안 0.2 초 동안 기다렸다가 alive 의 값을 0 으로 바꿔서 모
든 스레드가
 * 무한 루프를 빠져나올 수 있게 만든 후, 스레드가 자연스럽게 종료할 때까지 기다리고
메인을 종료한다.
 */
int main(void)
{
    int i;
    int rarg[RNUM], warg[WNUM];
    pthread_t rthid[RNUM];
    pthread_t wthid[WNUM];
    struct timespec req, rem;
    pthread_mutex_init(&mutex_lock, NULL);
    pthread_mutex_init(&reader_lock, NULL);
    pthread_mutex_init(&access_lock, NULL);

    /*
     * Create RNUM reader threads
     */
    for (i = 0; i < RNUM; ++i) {
        rarg[i] = i;
        if (pthread_create(rthid+i, NULL, reader, rarg+i) != 0) {
            fprintf(stderr, "pthread_create error\n");
            exit(-1);
        }
    }
    /*
     * Create WNUM writer threads
     */
    for (i = 0; i < WNUM; ++i) {
        warg[i] = i;
        if (pthread_create(wthid+i, NULL, writer, warg+i) != 0) {
            fprintf(stderr, "pthread_create error\n");
            exit(-1);
        }
    }
    /*
     * Wait for 0.2 second while the threads are working
     */
    req.tv_sec = 0;
    req.tv_nsec = 200000000L;
    nanosleep(&req, &rem);
    /*
     * Now terminate all threads and leave
     */
    alive = 0;
```

```
    for (i = 0; i < RNUM; ++i)
        pthread_join(rthid[i], NULL);
    for (i = 0; i < WNUM; ++i)
        pthread_join(wthid[i], NULL);
    exit(0);
    pthread_mutex_destroy(&mutex_lock);
    pthread_mutex_destroy(&reader_lock);
    pthread_mutex_destroy(&access_lock);

}
```

프로그램 컴파일 과정

writer_prefer.c

```
> gcc -v writer_prefer.c -lpthread
Apple clang version 12.0.0 (clang-1200.0.32.27)
Target: x86_64-apple-darwin20.3.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
 "/Library/Developer/CommandLineTools/usr/bin/clang" -cc1 -triple x86_64-apple-macosx11.0.0 -Wdeprecated-objc-isa-usage -Werror=deprecated-objc-isa-usage -Werror=implicit-function
-declaration -emit-obj -mrelax-all -disable-free -disable-llvm-verifier -discard-value-names -main-file-name writer_prefer.c -mrelocation-model pic -pic-level 2 -mthread-model pos
ix -mframe-pointer=all -fno-strict-return -masm-verbose -munwind-tables -target-sdk-version=11.0 -target-cpu penryn -dwarf-column-info -debugger-tuning=lldb -target-linker-version
 609.6 -v -resource-dir /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0 -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk -I/usr/local/include -internal-isyst
em /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/local/include -internal-isystem /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/include -internal-externc-isyst
em /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include -internal-externc-isystem /Library/Developer/CommandLineTools/usr/include -Wno-reorder-init-list -Wno-implicit-i
nt-float-conversion -Wno-c99-designator -Wno-final-dtor-non-final-class -Wno-extra-semi-stmt -Wno-misleading-indentation -Wno-quoted-include-in-framework-header -Wno-implicit-fall
through -Wno-enum-enum-conversion -Wno-enum-float-conversion -fdebug-compilation-dir /Users/lys/desktop/univ/cpp -ferror-limit 19 -fmessage-length 179 -stack-protector 1 -fstack-c
heck -mdarwin-stkchk-strong-link -fblocks -fencode-extended-block-signature -fregister-global-dtors-with-atexit -fgnuc-version=4.2.1 -fobjc-runtime=macosx-11.0.0 -fmax-type-align=
16 -fdiagnostics-show-option -fcolor-diagnostics -o /var/folders/4v/gvsn66p55kb82v8gyphfgp_r0000gn/T/writer_prefer-afcf4f.o -x c writer_prefer.c
clang -cc1 version 12.0.0 (clang-1200.0.32.27) default target x86_64-apple-darwin20.3.0
ignoring nonexistent directory "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/local/include"
ignoring nonexistent directory "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/Library/Frameworks"
#include "..." search starts here:
#include <...> search starts here:
 /usr/local/include
 /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/include
 /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include
 /Library/Developer/CommandLineTools/usr/include
 /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/System/Library/Frameworks (framework directory)
End of search list.
 "/Library/Developer/CommandLineTools/usr/bin/ld" -demangle -lto_library /Library/Developer/CommandLineTools/usr/lib/libLTO.dylib -no_deduplicate -dynamic -arch x86_64 -platform_v
ersion macos 11.0.0 11.0 -syslibroot /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk -o a.out -L/usr/local/lib /var/folders/4v/gvsn66p55kb82v8gyphfgp_r0000gn/T/writer_prefer-a
fcf4f.o -lpthread -lSystem /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/lib/darwin/libclang_rt.osx.a
```

fair_reader_writer.c

```
> gcc -v fair_reader_writer.c -lpthread
Apple clang version 12.0.0 (clang-1200.0.32.27)
Target: x86_64-apple-darwin20.3.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
 "/Library/Developer/CommandLineTools/usr/bin/clang" -cc1 -triple x86_64-apple-macosx11.0.0 -Wdeprecated-objc-isa-usage -Werror=deprecated-objc-isa-usage -Werror=implicit-function
-declaration -emit-obj -mrelax-all -disable-free -disable-llvm-verifier -discard-value-names -main-file-name fair_reader_writer.c -mrelocation-model pic -pic-level 2 -mthread-mode
l posix -mframe-pointer=all -fno-strict-return -masm-verbose -munwind-tables -target-sdk-version=11.0 -target-cpu penryn -dwarf-column-info -debugger-tuning=lldb -target-linker-ve
rsion 609.6 -v -resource-dir /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0 -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk -I/usr/local/include -internal-
isystem /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/local/include -internal-isystem /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/include -internal-externc-
isystem /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include -internal-externc-isystem /Library/Developer/CommandLineTools/usr/include -Wno-reorder-init-list -Wno-impli
cit-int-float-conversion -Wno-c99-designator -Wno-final-dtor-non-final-class -Wno-extra-semi-stmt -Wno-misleading-indentation -Wno-quoted-include-in-framework-header -Wno-implicit
-fallthrough -Wno-enum-enum-conversion -Wno-enum-float-conversion -fdebug-compilation-dir /Users/lys/desktop/univ/cpp -ferror-limit 19 -fmessage-length 179 -stack-protector 1 -fst
ack-check -mdarwin-stkchk-strong-link -fblocks -fencode-extended-block-signature -fregister-global-dtors-with-atexit -fgnuc-version=4.2.1 -fobjc-runtime=macosx-11.0.0 -fmax-type-a
lign=16 -fdiagnostics-show-option -fcolor-diagnostics -o /var/folders/4v/gvsn66p55kb82v8gyphfgp_r0000gn/T/fair_reader_writer-f33323.o -x c fair_reader_writer.c
clang -cc1 version 12.0.0 (clang-1200.0.32.27) default target x86_64-apple-darwin20.3.0
ignoring nonexistent directory "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/local/include"
ignoring nonexistent directory "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/Library/Frameworks"
#include "..." search starts here:
#include <...> search starts here:
 /usr/local/include
 /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/include
 /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include
 /Library/Developer/CommandLineTools/usr/include
 /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/System/Library/Frameworks (framework directory)
End of search list.
 "/Library/Developer/CommandLineTools/usr/bin/ld" -demangle -lto_library /Library/Developer/CommandLineTools/usr/lib/libLTO.dylib -no_deduplicate -dynamic -arch x86_64 -platform_v
ersion macos 11.0.0 11.0 -syslibroot /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk -o a.out -L/usr/local/lib /var/folders/4v/gvsn66p55kb82v8gyphfgp_r0000gn/T/fair_reader_wri
ter-f33323.o -lpthread -lSystem /Library/Developer/CommandLineTools/usr/lib/clang/12.0.0/lib/darwin/libclang_rt.osx.a
```

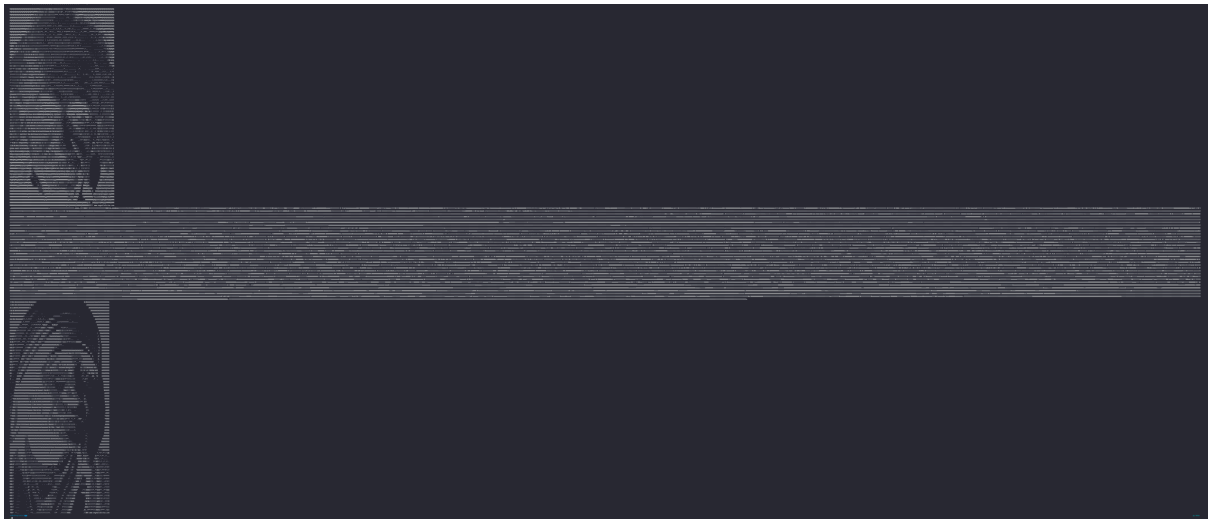
실행 결과물 및 설명

writer_prefer.c

writer가 먼저 하나씩 실행된 후, reader가 겹치면서 동시에 출력된다.

소스코드 상 reader를 실행하는 쓰레드가 먼저 만들어지기 때문에 reader가 출력되다가 writer가 들어오면 writer를 우선으로 출력하고 남은 reader들이 마지막에 출력된다.

fair_reader_writer.c



reader와 writer 먼저 들어온 순서대로 출력된다.

writer -> reader -> writer -> writer -> reader -> writer 순으로 출력되는 모습이다.