mycp, myls, myshell

Index

❖ mycp

❖ mycp_adv
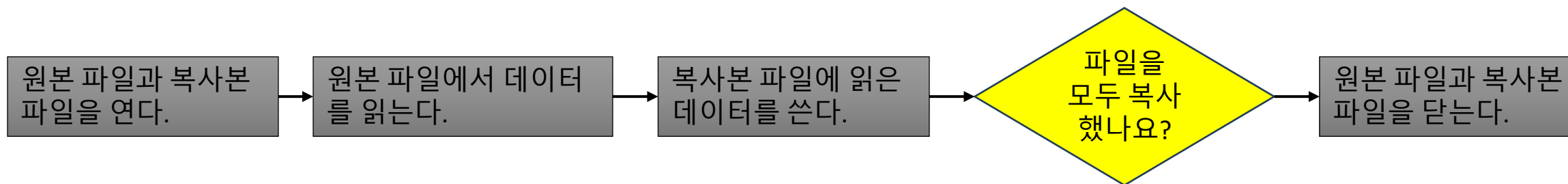
❖ myls

❖ Semaphore

❖ Dead lock

❖ Starvation

mycp

❖ Input

▪ Usage: ./mycp [*origin file*] [*destination file*]

❖ Output

▪ 원본 파일 origin file 의 user data가 적혀 있는 복사본 파일 destination file 생성

❖ mycp algorithm

| 원본 파일과 복사본 파일을 연다. | → | 원본 파일에서 데이터를 읽는다. | → | 복사본 파일에 읽은 데이터를 쓴다. | → | 파일을 모두 복사 했나요? | → | 원본 파일과 복사본 파일을 닫는다. |

mycp
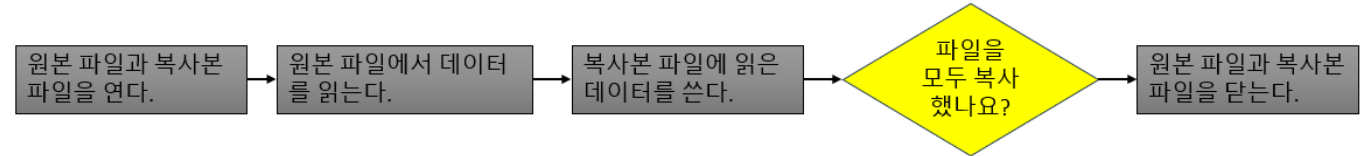
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <errno.h>
6  #define MAX_BUF 64
7
8  int main(int argc, char *argv[]){
9      int fd_origin, fd_dest, read_size, write_size =0;
10     char buf[MAX_BUF];
11
12     if (argc!=3){
13         printf("USAGE: %s origin dest\n",argv[0]);
14         exit(-1);
15     }
16
17     fd_origin = [1] fill out here using system call     ;
18     fd_dest = [1-1] fill out here using system call                    ;
19     if (fd_origin < 0 || fd_dest <0){
20         //open error handling
21         perror("fd open error\n");
22     }
23
24 //read from the origin file
25     while((read_size = [2] fill out here using system call              ) != 0 ){
26         //write to the dest file
27         write_size = [3] fill out here using system call       ;
28     }
29
30
31     [4] fill out here using system call
32 }
```

❖ mycp algorithm

| 원본 파일과 복사본 파일을 연다. | 원본 파일에서 데이터를 읽는다. | 복사본 파일에 읽은 데이터를 쓴다. | 파일을 모두 복사 했나요? | 원본 파일과 복사본 파일을 닫는다. |

## Using System call

```
open

read

write

close
```

mycp

❖ mycp 실행화면

- cat [file]

- 파일의 데이터 비교

```
[ec2-user@ip-172-31-15-105 taba9]$ ls -l
total 28
-rwxr-xr-x. 1 ec2-user ec2-user 17744 Aug 27 17:49 mycp
-rw-r--r--. 1 ec2-user ec2-user   766 Aug 27 18:06 mycp.c
-rw-r--r--. 1 ec2-user ec2-user    17 Aug 27 18:24 origin
[ec2-user@ip-172-31-15-105 taba9]$ ./mycp origin dest
[ec2-user@ip-172-31-15-105 taba9]$ cat dest
I am origin file
[ec2-user@ip-172-31-15-105 taba9]$ cat origin
I am origin file
[ec2-user@ip-172-31-15-105 taba9]$
```

❖ 파일 file

- User data

  - 실제 파일에 저장되는 데이터

- Metadata

  - File의 정보와 user data의 위치를 가리키는 데이터

  - 파일 유형, 파일 크기, 파일 권한…

Medata for file

file

inode

파일 유형

파일 크기

파일 권한

…
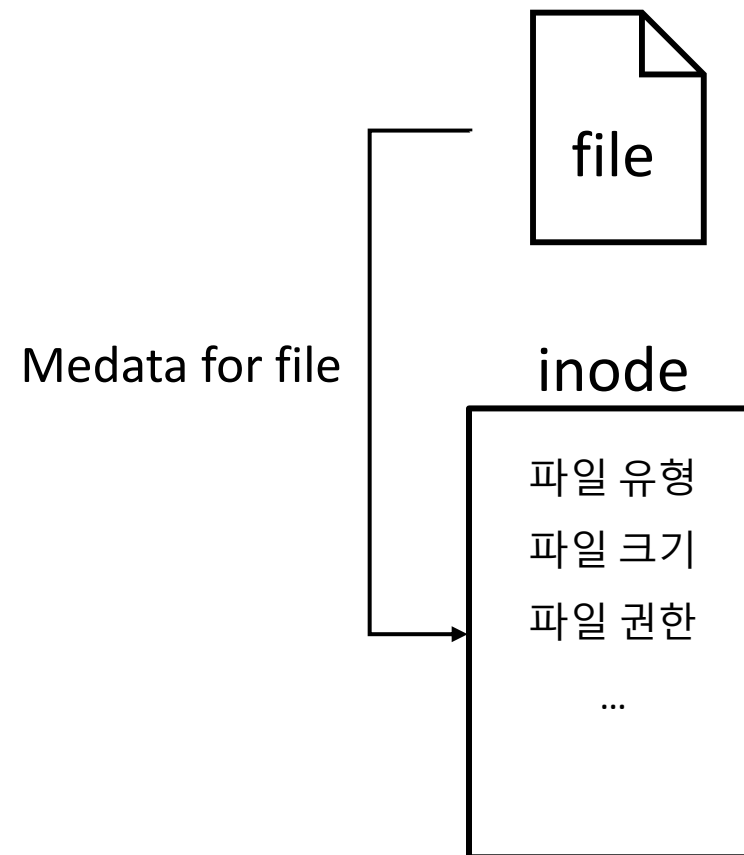
mycp_adv

❖ Input

- Usage: ./mycp_adv [*origin file*] [*destination file*]

❖ Output

- 원본 파일 origin file 의 user data가 적혀 있는 복사본 파일 destination file 생성

- 파일의 속성 정보 (Metadata)를 포함한 완전 복사

mycp_adv

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <errno.h>
6  #include <sys/stat.h>
7  #define MAX_BUF 64
8
9  int main(int argc, char *argv[]){
10     int fd_origin, fd_dest, read_size, write_size = 0;
11     char buf[MAX_BUF];
12     struct stat* stat_origin = (struct stat*)malloc(sizeof(struct stat));
13     mode_t flag_origin;
14
15     if (argc != 3){
16         printf("USAGE: %s origin dest\n", argv[0]);
17         exit(-1);
18     }
19     fd_origin = open(argv[1], O_RDONLY);
20
21     [1] get file attribute structure from fd_origin
22
23     flag_origin = stat_origin->   [2] let's get member from struct stat "stat_origin->field_here
24
25     fd_dest = open(argv[2], O_RDWR | O_CREAT | O_EXCL | O_SYNC, flag_origin);
26  while((read_size = read(fd_origin, buf, MAX_BUF)) != 0){
27         write_size = write(fd_dest, buf, read_size);
28     }
29     close(fd_origin);
30     close(fd_dest);
31 }
```

Using System call

fstat

mycp_adv

❖ mycp_adv 실행화면

- cat [file]

- 파일의 데이터 비교

```
[ec2-user@ip-172-31-15-105 taba9]$ ./mycp_adv origin dest
[ec2-user@ip-172-31-15-105 taba9]$ ls -l
total 56
-rw-r--r--. 1 ec2-user ec2-user    17 Aug 27 20:23 dest
-rwxr-xr-x. 1 ec2-user ec2-user 17744 Aug 27 17:49 mycp
-rwxr-xr-x. 1 ec2-user ec2-user 17792 Aug 27 20:23 mycp_adv
-rw-r--r--. 1 ec2-user ec2-user   805 Aug 27 20:18 mycp_adv.c
-rw-r--r--. 1 ec2-user ec2-user   766 Aug 27 18:06 mycp.c
-rw-r--r--. 1 ec2-user ec2-user    17 Aug 27 18:24 origin
[ec2-user@ip-172-31-15-105 taba9]$ cat origin
I am origin file
[ec2-user@ip-172-31-15-105 taba9]$ cat dest
I am origin file
```

mycp-adv

❖ stat [file]

- 파일의 메타데이터 비교

```
[ec2-user@ip-172-31-15-105 taba9]$ stat origin
  File: origin
  Size: 17              Blocks: 8          IO Block: 4096    regular file
Device: 10302h/66306d   Inode: 8413986     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/ec2-user)   Gid: ( 1000/ec2-user)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2024-08-27 18:24:23.238560580 +0000
Modify: 2024-08-27 18:24:12.954589960 +0000
Change: 2024-08-27 18:24:12.954589960 +0000
 Birth: 2024-08-27 18:24:12.954589960 +0000
[ec2-user@ip-172-31-15-105 taba9]$ stat dest
  File: dest
  Size: 17              Blocks: 8          IO Block: 4096    regular file
Device: 10302h/66306d   Inode: 8413988     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/ec2-user)   Gid: ( 1000/ec2-user)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2024-08-27 20:24:06.411915356 +0000
Modify: 2024-08-27 20:23:42.476984046 +0000
Change: 2024-08-27 20:23:42.476984046 +0000
 Birth: 2024-08-27 20:23:42.476984046 +0000
```

myls

❖ Input

- Usage: ./myls *[argument]*   *선택

- [argument]는 디렉토리

❖ Output

- 디렉토리 내의 파일과 디렉토리 목록을 보여줌

- 1. ./myls

  • 현재 디렉토리 내의 파일과 디렉토리 목록을 보여줌

- 2. ./myls [argument]

  • argument에 해당하는 디렉토리 내의 파일과 디렉토리 목록을 보여줌

myls

```c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <errno.h>
6  #include <dirent.h>
7  #include <sys/types.h>
8
9  int main(int argc, char *argv[]) {
10 // 변 수 선 언
11     DIR *dir = NULL;
12     struct dirent* dentry = NULL;
13     char *dir_name = ".";
14
15     // 예 외 처 리
16     if (argc == 1) { // args 없 는 경 우 현 재 디 렉 토 리 "." 내 용 을 보 여 줌 .
17         dir = opendir(dir_name);
18     } else if (argc == 2) {
19         dir_name = argv[1]; // 디 렉 토 리 이 름 을 argv에 서 가 져 옴
20         dir = [1] fill out here using directory syscall
21     } else {
22         printf("argc %d : We only accept 1 or 2 args for now\n", argc);
23         exit(-1);
24     }
25     while ((dentry = [2] using dir syscall ) != NULL) {
26         printf("%s \n", dentry->d_name); // 디 렉 토 리 항 목 이 름 을 출 력
27     }
28     [3] using dir syscall
29 }
```

Using System call

opendir
readdir
closedir

myls

❖ myls 실행화면

- 인자(argv) 0개와 1개 비교

```
[ec2-user@ip-172-31-15-105 taba9]$ ./myls
.
..
mycp
mycp.c
origin
mycp_adv.c
mycp_adv
dest
myls.c
myls
```

```
[ec2-user@ip-172-31-15-105 taba9]$ ./myls ..
.
..
.bash_logout
.bash_profile
.ssh
.bash_history
.vimrc
.bashrc
day2
taba7
taba9
.viminfo
```

❖ create()

❖ mkdir(), readdir(), rmdir()

❖ pipe()

❖ mknod()

❖ link(),unlink()

❖ dup(),dup2()

❖ stat(),fstat()

❖ chmod(), fchmod()

❖ Ioctl(), fcntl()

❖ Sync(), fsync()

binary semaphore

❖ thread_bin_sem.c

(1) sem_init(semaphore, p_shared, initial_value): 세마포어를 초기화 및 생성

(2) sem_destroy(semaphore): 세마포어를 제거

```c
 1  #include <stdlib.h>
 2  #include <unistd.h>
 3  #include <stdio.h>
 4  #include <assert.h>
 5  #include <pthread.h>
 6  #include <semaphore.h>
 7
 8  int count = 0;
 9  int nthread = 1;
10  int worker_loop_cnt = 1;
11  sem_t semaphore;
12  static void *work(void *num);

14  int main(int argc, char *argv[]){
15      pthread_t *th;
16      void *value;
17      long i;
18
19      if(argc < 3){
20          fprintf(stderr, "%s parameter : nthread, worker_loop_cnt\n", argv[0]);
21          exit(-1);
22      }
23
24      nthread = atoi(argv[1]);
25      worker_loop_cnt = atoi(argv[2]);
26      th = malloc(sizeof(pthread_t) * nthread);
27
28  (1) sem_init(&semaphore, 0, 1);
29      printf("main: begin (count = %d)\n", count);
30      for(i = 0; i < nthread; i++)
31          assert(pthread_create(&th[i], NULL, work, (void*)i) == 0);
32      for(i = 0; i < nthread; i++)
33          assert(pthread_join(th[i], &value) == 0);
34  (2) sem_destroy(&semaphore);
35      printf("main: done (count = %d)\n", count);
36  }
```

binary semaphore

❖ thread_bin_sem.c

(1) sem_wait(): semaphore 값 감소

(2) sem_post(): semaphore 값 증가

```
38  static void *work(void *num) {
39      long number = (long)num;
40
41  (1) sem_wait(&semaphore);
42
43      for(int i = 0; i < worker_loop_cnt; i++)
44          count++;
45      printf("Thread number %d: %d \n", number, count);
46
47  (2) sem_post(&semaphore);
48
49      return NULL;
50  }
```

binary semaphore

❖ 실행파일

```
[root@ip-172-31-15-105 semaphore]# gcc thread_bin_sem.c -lpthread -o thread_bin_sem.out
```

❖결과

```
[root@ip-172-31-15-105 semaphore]# ./thread_bin_sem.out 5 10000
main: begin (count = 0)
Thread number 0: 10000
Thread number 3: 20000
Thread number 2: 30000
Thread number 1: 40000
Thread number 4: 50000
main: done (count = 50000)
```

counting semaphore

❖ thread_counting_sem.c

▪ 선언부

```c
 1 #include <stdlib.h>
 2 #include <unistd.h>
 3 #include <stdio.h>
 4 #include <assert.h>
 5 #include <pthread.h>
 6 #include <semaphore.h>
 7
 8 #define SEM_COUNT 3
 9
10 int count[SEM_COUNT];
11 int working [SEM_COUNT];
12 int nthread = 1;
13 int worker_loop_cnt = 1;
14
15 pthread_mutex_t lock;
16 sem_t semaphore;
17
18 static void *work(void *num);
19
```

counting semaphore

❖ thread_counting_sem.c

▪ main 함수

```c
20 int main(int argc, char *argv[])
21 {
22     pthread_t *th;
23     void *value;
24     long i;
25
26     if (argc < 3) {
27         fprintf(stderr, "%s parameter : nthread, worker_loop_cnt\n", argv[0]);
28         exit(-1);
29     }
30
31     nthread = atoi(argv[1]);
32     worker_loop_cnt = atoi(argv[2]);
33
34     th = malloc(sizeof(pthread_t) * nthread);
35
36     pthread_mutex_init(&lock, NULL); // initialize the lock
37     sem_init(&semaphore, 0, SEM_COUNT); // init sem
38
39     for(i = 0; i < nthread; i++) {
40         assert(pthread_create(&th[i], NULL, work, (void*) i) == 0);
41     }
42
43     for(i = 0; i < nthread; i++) {
44         assert(pthread_join(th[i], &value) == 0);
45     }
47         sem_destroy(&semaphore);
48
49         free(th);
50
51         printf("Count array :\n");
52         for(int i = 0; i < SEM_COUNT; i++) {
53             printf("%d ", count[i]);
54         }
55
56         printf("\nComplete\n");
57 }
```

counting semaphore

❖ thread_counting_sem.c

- work 함수

```
59  static void *work(void* num)
60  {
61      long number = (long)num;
62      int count_index = -1;
63  
64      sem_wait(&semaphore); // sem count down
65  
66      pthread_mutex_lock(&lock); // lock
67      for(int i = 0; i < SEM_COUNT; i++) {
68          if(working[i] == 0) {
69              working[i] = 1;
70              count_index = i;
71              break;
72          }
74      pthread_mutex_unlock(&lock); // unlock
75  
76      if(count_index == -1){
77          fprintf(stderr, "Thread number %d: count_index < 0\n", number);
78          exit(-1);
79      }
80  
81      for(int i = 0; i < worker_loop_cnt; i++)
82          count[count_index]++;
83  
84      //printf("Thread number %d: %d \n", number, count[count_index]);
85  
86      pthread_mutex_lock(&lock); // lock
87      working[count_index] = 0;
88      pthread_mutex_unlock(&lock); // unlock
89  
90      sem_post(&semaphore); // sem count up
91  
92      return NULL;
93  }
```

`[root@ip-172-31-15-105 semaphore]# gcc thread_counting_sem.c -lpthread -o thread_counting_sem.out`

Dead lock

❖ thread_2lock.c

```
 1  #include <stdlib.h>
 2  #include <unistd.h>
 3  #include <stdio.h>
 4  #include <assert.h>
 5  #include <pthread.h>
 6
 7  int first_count = 0;
 8  int second_count = 0;
 9  int nthread = 1;
10  int nthread_one = 1;
11
12  int main_loop_cnt = 1;
13
14  pthread_mutex_t first_lock;
15  pthread_mutex_t second_lock;
```

Dead lock

❖ thread_2lock.c

```c
59 int main(int argc, char *argv[])
60 {
61     pthread_t *th;
62     void *value;
63     long i;
64
65     if (argc < 3) {
66         fprintf(stderr, "%s parameter : nthread, main_loop_cnt\n", argv[0]);
67         exit(-1);
68     }
69
70     nthread = atoi(argv[1]);
71     nthread_one = nthread/2;
72
73     main_loop_cnt = atoi(argv[2]);
74
75     th = malloc(sizeof(pthread_t) * nthread);
76
77     pthread_mutex_init(&first_lock, NULL); // initialize the lock
78     pthread_mutex_init(&second_lock, NULL); // initialize the lock
79
80     for(int loop = 0; loop < main_loop_cnt; loop++){
81         printf("---- loop %d ----\n", loop);
82
83         for(i = 0; i < nthread_one; i++)
84             assert(pthread_create(&th[i], NULL, work_one, (void*) i) == 0);
85
86         for(i = nthread_one; i < nthread; i++)
87             assert(pthread_create(&th[i], NULL, work_two, (void*) i) == 0);
88
89         for(i = 0; i < nthread; i++)
90             assert(pthread_join(th[i], &value) == 0);
91
92         first_count = 0;
93         second_count = 0;
94     }
95 }
```

Computer Security & OS LAB

Dead lock

❖ thread_2lock.c

```
[root@ip-172-31-15-105 semaphore]# gcc thread deadlock.c -lpthread -o thread deadlock.out
```

```c
17 static void *work_one(void* num)
18 {
19     long number = (long)num;
20     int answer = 0;
21  pthread_mutex_lock(&first_lock); // lock
22     pthread_mutex_lock(&second_lock); // lock
23
24     answer = first_count + second_count;
25
26     printf("Work_one : %d \n", answer);
27
28     first_count++;
29     second_count++;
30
31     pthread_mutex_unlock(&second_lock); // unlock
32     pthread_mutex_unlock(&first_lock); // unlock
33
34     return NULL;
35 }
```

```c
37 static void *work_two(void* num)
38 {
39     long number = (long)num;
40
41     int answer = 0;
42
43     pthread_mutex_lock(&second_lock); // lock
44     pthread_mutex_lock(&first_lock); // lock
45
46     answer = first_count + second_count;
47
48     printf("Work_two : %d \n", answer);
49
50     first_count++;
51     second_count++;
52
53     pthread_mutex_unlock(&first_lock); // unlock
54     pthread_mutex_unlock(&second_lock); // unlock
55
56     return NULL;
57 }
```

Lock: Starvation

❖ thread_2lock.c

```c
1 #include <stdlib.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <assert.h>
5 #include <pthread.h>
6 #include <semaphore.h>
7
8 sem_t mutex;
9 int count = 0;
10 int nthread = 1;
11 int worker_loop_cnt = 1;
```

Lock: Starvation

❖ thread_2lock.c

```
13 void *work(void *number)
14 {
15     int i;
16     long thread_id = (long)number;
17
18     for (i = 0; i < worker_loop_cnt; i++) {
19         sem_wait(&mutex);
20         count++;
21         printf("Thread %d: count = %d\n", thread_id, count);
22         sem_post(&mutex);
23     }
24
25     pthread_exit(NULL);
26 }
```

Lock: Starvation

❖ thread_2lock.c

```c
28 int main(int argc, char *argv[])
29 {
30     pthread_t *th;
31     void *value;
32     long i;
33
34     if(argc < 3){
35         fprintf(stderr, "%s parameter : nthread, worker_loop_cnt\n", argv[0]);
36         exit(-1);
37     }
38     nthread = atoi(argv[1]);
39     worker_loop_cnt = atoi(argv[2]);
40
41     th = malloc(nthread * sizeof(pthread_t));
42     if (th == NULL) {
43         fprintf(stderr, "Memory allocation failed\n");
44         exit(-1);
45     }
46
47     sem_init(&mutex, 0, 1);
48
49     for (i = 0; i < nthread; i++) {
50         long *arg = malloc(sizeof(*arg));
51         *arg = i;
52         assert(pthread_create(&th[i], NULL, work, arg) == 0);
53     }
54
55     for (i = 0; i < nthread; i++)
56         assert(pthread_join(th[i], &value) == 0);
57
58     sem_destroy(&mutex);
59     free(th);
60
61     return 0;
62 }
```

Lock: Starvation

❖ 실행파일

```
[root@ip-172-31-15-105 semaphore]# gcc starvation.c -lpthread -o starvation.out
```

❖ 결과

```
[root@ip-172-31-15-105 semaphore]# ./starvation.out 5 3
Thread 23532240: count = 1
Thread 23532240: count = 2
Thread 23532240: count = 3
Thread 23532560: count = 4
Thread 23532560: count = 5
Thread 23532560: count = 6
Thread 23532880: count = 7
Thread 23532880: count = 8
Thread 23532880: count = 9
Thread 23533520: count = 10
Thread 23533520: count = 11
Thread 23533520: count = 12
Thread 23533200: count = 13
Thread 23533200: count = 14
Thread 23533200: count = 15
```