

106119029 Lab 4 Report, OS LAB, Shared Memory

Question

Write a program with a producer and two consumers. The producer puts an integer array in the shared memory, and the cons1 adds the elements of the array. Cons2 finds the standard deviation.

- The way I have done this is using one shared memory object which i create and fill in producer.c and use the same thing in both consumer1.c and consumer2.c. However I did not delete the shared memory in any of these files knowingly. I did not delete because I wanted to not run `./prod` 2 times to get run both the cons files. I have set up the `Makefile` in such a way that it will run the `shm_unlink.c` file when i run `make clean`.
- I could have done this using two different shared memory for `consumer1.c` and `consumer2.c` and deleted them there itself after the usage was complete. But had I done that, I would not be able to call `cons1` and `cons2` second time without rerunning `prod`. So thats a drawback of doing that. With single shared memory I can run `cons1` and `cons2` as many times as I want because I dont delete them after they are used. Only after I run `make clean`, it will delete the shared memory.
- I open a shared file descriptor in producer.c called "array" with `shm_open` and then map that file descriptor to some memory using `mmap` call with size `PAGE_SIZE(4096)`. I then put data 0 through 1023 in that memory which will be reflected on the shared file. I have printed before and after case for the memory. I then unmap the memory with `munmap` and also close the file descriptor.
- I open the shared memory again in consumer1.c and map it to some heap memory with `mmap` and then find the sum of the data inside the shared memory by treating it as int array. I did the same for consumer2.c but there I found the standard deviation of the array content.
- The file `unlink_mem.c` is responsible for deleting the shared memory object created by the producer and will be executed when I run `make clean`

Code

- producer.c

```
#include <fcntl.h>
#include <stdio.h>
#include <sys/mman.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <unistd.h>
#define PAGE_SIZE 4096
int main() {
```

```

// creates if it doesnt exist
int shm_fd = shm_open("/array", O_CREAT | O_RDWR | O_EXCL, 0666);
if (shm_fd == -1) {
    printf("Failed to create or the object already exist\n");
    exit(1);
}
ftruncate(shm_fd, PAGE_SIZE);
// shared memory is mapped to some heap memory with mmap, it has READ and
// WRITE privileges It is shared as well indicated by MAP_SHARED
int *arr=
    mmap(NULL, PAGE_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
if (arr == MAP_FAILED) {
    printf("mmap failed\n");
    exit(2);
}
printf("BEFORE:\n");
for (int i = 0; i < (PAGE_SIZE / sizeof(int)); i++)
    printf("%x ", arr[i]);
for (int i = 0; i < PAGE_SIZE / sizeof(int); i++)
    arr[i] = i;
printf("\nAFTER:\n");
for (int i = 0; i < (PAGE_SIZE / sizeof(int)); i++)
    printf("%x ", arr[i]);
printf("\n");
close(shm_fd);
munmap(arr, PAGE_SIZE);
}

```

- consumer1.c

```

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <unistd.h>

#define PAGE_SIZE 4096

int main() {
    printf("THE SUM OF ALL THE ELEMENTS: ");
    int fd = shm_open("/array", O_RDONLY, 0444);
    if (fd == -1) {
        printf("Couldnt open shared memory\n");
        exit(1);
    }
}

```

```

int *arr = mmap(NULL, PAGE_SIZE, PROT_READ, MAP_SHARED, fd, 0);
if (arr == MAP_FAILED) {
    printf("mmap Failed\n");
    exit(2);
}
int sum = 0;
for (int i = 0; i < PAGE_SIZE / sizeof(int); i++) {
    sum += arr[i];
}
printf("%d\n", sum);
}

```

- consumer2.c

```

#include <fcntl.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <unistd.h>
#define PAGE_SIZE 4096
int main() {
    int fd = shm_open("/array", O_RDONLY, 0444);
    if (fd == -1) {
        printf("Couldnt open shared memory\n");
        exit(1);
    }
    int *arr = mmap(NULL, PAGE_SIZE, PROT_READ, MAP_SHARED, fd, 0);
    if (arr == MAP_FAILED) {
        printf("mmap Failed\n");
        exit(2);
    }
    int sum = 0;
    double mean, standard_dev_numerator = 0, standard_dev;
    for (int i = 0; i < PAGE_SIZE / sizeof(int); i++) {
        sum += arr[i];
    }
    mean = ((double)sum) / (double)(PAGE_SIZE / sizeof(int));
    printf("THE MEAN IS %f\n", mean);
    for (int i = 0; i < PAGE_SIZE / sizeof(int); i++) {
        standard_dev_numerator += pow((double)arr[i] - mean, 2);
    }
    printf("THE numerator IS %f\n", standard_dev_numerator);
    standard_dev =
        sqrt((standard_dev_numerator / (double)(PAGE_SIZE / sizeof(int))));
}

```

```

printf("THE STANDARD DEVIATION IS : %f\n", standard_dev);
close(fd);
munmap(arr, PAGE_SIZE);
}

```

- unlink_mem.c

```

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <unistd.h>

int main() { shm_unlink("/array"); }

```

- Makefile

```

CFLAGS=-lm -lrt
CC=gcc

all: prod cons1 cons2

prod: producer.c
    $(CC) $(CFLAGS) -o prod producer.c

cons1: prod consumer1.c
    $(CC) $(CFLAGS) -o cons1 consumer1.c

cons2: prod consumer2.c
    $(CC) $(CFLAGS) -o cons2 consumer2.c

unlink_shm: unlink_mem.c prod
    $(CC) $(CFLAGS) -o unlink_shm unlink_mem.c

clean: unlink_shm
    rm prod cons1 cons2
    ./unlink_shm
    rm ./unlink_shm

```

Output

- prod

```
gcc -m-lrt -o prod producer.c
gcc -m-lrt -o cons consumer.c
gcc -m-lrt -o cons2 consumer2.c
gcc -S./cm4/CSML4-OSLab/Lab4 -x ./prod
```

BEFORE:

AFTER:

- cons1 and cons2

```

λ ~/Acads/Sem4/CSLR42-OSLab/Lab4 → ./cons1
THE SUM OF ALL THE ELEMENTS: 523776
λ ~/Acads/Sem4/CSLR42-OSLab/Lab4 → ./cons2
THE MEAN IS 511.500000
THE numerator IS 89478400.000000
THE STANDARD DEVIATION IS : 295.603197
λ ~/Acads/Sem4/CSLR42-OSLab/Lab4 → make clean
gcc -lm -lrt -o unlink_shm unlink_mem.c
rm prod cons1 cons2
./unlink_shm
rm ./unlink_shm
λ ~/Acads/Sem4/CSLR42-OSLab/Lab4 → ls
demofiles ss consumer1.c consumer2.c Makefile producer.c report.md report.pdf unlink_mem.c
λ ~/Acads/Sem4/CSLR42-OSLab/Lab4 → ls /dev/shm/
λ ~/Acads/Sem4/CSLR42-OSLab/Lab4 →

```

Code screenshots

- producer.c

```

1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <sys/mman.h>
5 #include <sys/shm.h>
6 #include <sys/stat.h>
7 #include <unistd.h>
8 #define PAGE_SIZE 4096
9 int main() {
10     // creates if it doesnt exist
11     int shm_fd = shm_open("/array", O_CREAT | O_RDWR | O_EXCL, 0666);
12     if (shm_fd == -1) {
13         printf("Failed to create or the object already exist\n");
14         exit(1);
15     }
16     ftruncate(shm_fd, PAGE_SIZE);
17     // shared memory is mapped to some heap memory with mmap, it has READ and
18     // WRITE privileges It is shared as well indicated by MAP_SHARED
19     int *arr =
20     mmap(NULL, PAGE_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
21     if (arr == MAP_FAILED) {
22         printf("mmap failed\n");
23         exit(2);
24     }
25     printf("BEFORE:\n");
26     for (int i = 0; i < (PAGE_SIZE / sizeof(int)); i++)
27         printf("%x ", arr[i]);
28     for (int i = 0; i < PAGE_SIZE / sizeof(int); i++)
29         arr[i] = i;
30     printf("\nAFTER:\n");
31     for (int i = 0; i < (PAGE_SIZE / sizeof(int)); i++)
32         printf("%x ", arr[i]);
33     printf("\n");
34     close(shm_fd);
35     munmap(arr, PAGE_SIZE);
36 }

```

- consumer1.c

```

15 #include <fcntl.h>
14 #include <stdio.h>
13 #include <stdlib.h>
12 #include <sys/mman.h>
11 #include <sys/shm.h>
10 #include <sys/stat.h>
9 #include <unistd.h>
8
7 #define PAGE_SIZE 4096
6
5 int main() {
4     printf("THE SUM OF ALL THE ELEMENTS: ");
3     int fd = shm_open("/array", O_RDONLY, 0444);
2     if (fd == -1) {
1     printf("Couldnt open shared memory\n");
16     exit(1);
1     }
2     int *arr = mmap(NULL, PAGE_SIZE, PROT_READ, MAP_SHARED, fd, 0);
3     if (arr == MAP_FAILED) {
4     printf("mmap Failed\n");
5     exit(2);
6     }
7     int sum = 0;
8     for (int i = 0; i < PAGE_SIZE / sizeof(int); i++) {
9     sum += arr[i];
10    }
11    printf("%d\n", sum);
12 }

```

- consumer2.c

```

9 #include <fcntl.h>
10 #include <math.h>
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <sys/mman.h>
14 #include <sys/shm.h>
15 #include <sys/stat.h>
16 #include <unistd.h>
17 #define PAGE_SIZE 4096
18 int main() {
19     int fd = shm_open("/array", O_RDONLY, 0444);
20     if (fd == -1) {
21         printf("Couldnt open shared memory\n");
22         exit(1);
23     }
24     int *arr = mmap(NULL, PAGE_SIZE, PROT_READ, MAP_SHARED, fd, 0);
25     if (arr == MAP_FAILED) {
26         printf("mmap Failed\n");
27         exit(2);
28     }
29     int sum = 0;
30     double mean, standard_dev_numerator = 0, standard_dev;
31     for (int i = 0; i < PAGE_SIZE / sizeof(int); i++) {
32         sum += arr[i];
33     }
34     mean = ((double)sum) / (double)(PAGE_SIZE / sizeof(int));
35     printf("THE MEAN IS %f\n", mean);
36     for (int i = 0; i < PAGE_SIZE / sizeof(int); i++) {
37         standard_dev_numerator += pow((double)arr[i] - mean, 2);
38     }
39     printf("THE numerator IS %f\n", standard_dev_numerator);
40     standard_dev =
41         sqrt((standard_dev_numerator / (double)(PAGE_SIZE / sizeof(int))));
42     printf("THE STANDARD DEVIATION IS : %f\n", standard_dev);
43     close(fd);
44     munmap(arr, PAGE_SIZE);
45 }

```

- Makefile

```

1 CFLAGS=-lm -lrt
2 CC=gcc
3
4 all: prod cons1 cons2
5
6 prod: producer.c
7     $(CC) $(CFLAGS) -o prod producer.c
8
9 cons1: prod consumer1.c
10    $(CC) $(CFLAGS) -o cons1 consumer1.c
11
12 cons2: prod consumer2.c
13    $(CC) $(CFLAGS) -o cons2 consumer2.c
14
15 unlink_shm: unlink_mem.c prod
16    $(CC) $(CFLAGS) -o unlink_shm unlink_mem.c
17
18 clean: unlink_shm
19     rm prod cons1 cons2
20     ./unlink_shm
21     rm ./unlink_shm

```