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 itslf 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 priviliges 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"); }
```

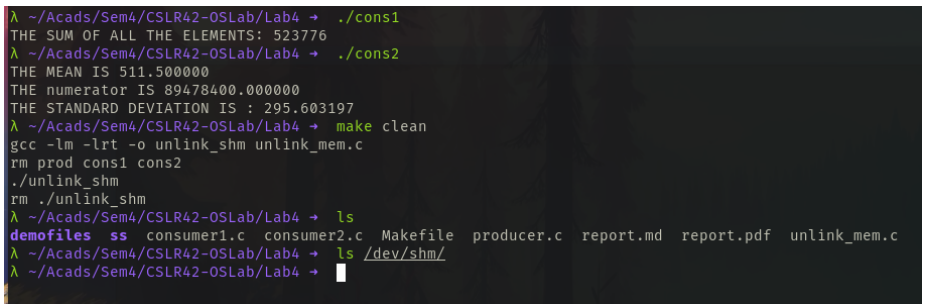
- Makefile

```
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

- cons1 and cons2



Code screenshots

- producer.c

```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() {
  // 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 priviliges 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

```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

```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);
}
```

- 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
```