Họ và tên: Nguyễn Trọng Đạt

MSSV: 52100176

Lớp: 21050301

Lab 5_3

Bài 1:

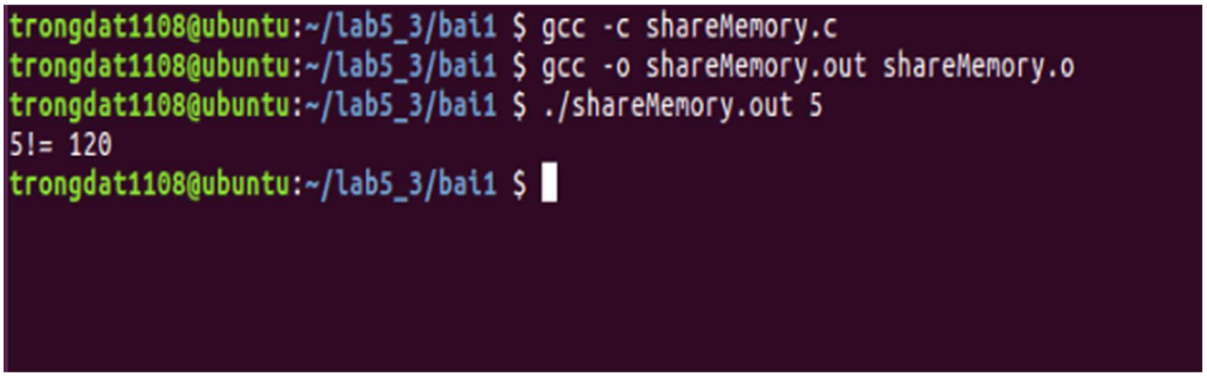
Share Memory:

```
#include <stdio.h>
#include <unistd.h>
#include <limits.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define SIZE 256
int main(int argc, char* argv[])
{
	int *shm, shmid, k,pid;
	key_t key;
	if((key=ftok(".",65))==-1){
		perror("Key created.\n");
		return 1;
```

```
 }
 shmid = shmget(key, SIZE, IPC_CREAT | 0666);
 if (shmid == -1) {
        perror("Shared memory created.\n");
        return 2;
 }
 shm = (int*) shmat(shmid, 0, 0);
 pid = fork();
 if(pid==0) { // child
        shm[0] = atoi(argv[1]);
        sleep(4);
        printf ("%d!= %d\n", shm[0],shm[1]);
        shmdt((void*) shm);
        shmctl(shmid, IPC_RMID, (struct shmid_ds*) 0);
        return 0;
 }
 else if(pid >0) { // parent

        sleep(2);
        int i,cnt=1;
for(i=1;i<=shm[0];i++){
   cnt*=i;
}
shm[1]=cnt;
```

```
            shmdt((void*) shm);
            sleep(5);
            return 0;
        }
        else { perror("Fork failed."); return 4; }
        return 0;
}
```



Message queue:

Write.c

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>

struct mesg_buffer{
        long mesg_type;
        int mesg_num;
}message;

int main() {
        key_t key;
        int msgid;

        key = ftok("msg.txt", 1);

        msgid = msgget(key, 0666 | IPC_CREAT);
        message.mesg_type = 1;
        printf("Write Data: ");
        scanf("%d", &message.mesg_num);

        msgsnd(msgid, &message, sizeof(message), 0);

        msgrcv(msgid, &message, sizeof(message), 1, 0);

        printf("Data Received is: %d\n", message.mesg_num);
        msgctl(msgid, IPC_RMID, NULL);
        return 0;
}
```

Read.c

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>

struct mesg_buffer{
        long mesg_type;
        int mesg_num;
}message;

int main() {
        key_t key;
        int msgid;

        key = ftok("msg.txt", 1);

        msgid = msgget(key, 0666 | IPC_CREAT);

        msgrcv(msgid, &message, sizeof(message), 1, 0);

        int i, fact = 1;
        for(i = 1; i <= message.mesg_num; i++){
                fact = fact*i;
        }
        message.mesg_num = fact;
        msgsnd(msgid, &message, sizeof(message), 0);
        msgctl(msgid, IPC_RMID, NULL);
        return 0;
}
```

```
trongdat1108@ubuntu:~/lab5_3/bai1 $ gcc -c write.c
trongdat1108@ubuntu:~/lab5_3/bai1 $ gcc -o write.out write.o
trongdat1108@ubuntu:~/lab5_3/bai1 $ ./write.out
Write Data: 5
Data Received is: 120
trongdat1108@ubuntu:~/lab5_3/bai1 $
```

```
trongdat1108@ubuntu: ~/lab5_3/bai1
trongdat1108@ubuntu:~/lab5_3/bai1 $ gcc -c read.c
trongdat1108@ubuntu:~/lab5_3/bai1 $ gcc -o read.out read.o
trongdat1108@ubuntu:~/lab5_3/bai1 $ ./read.out
trongdat1108@ubuntu:~/lab5_3/bai1 $
```

Bài 2:

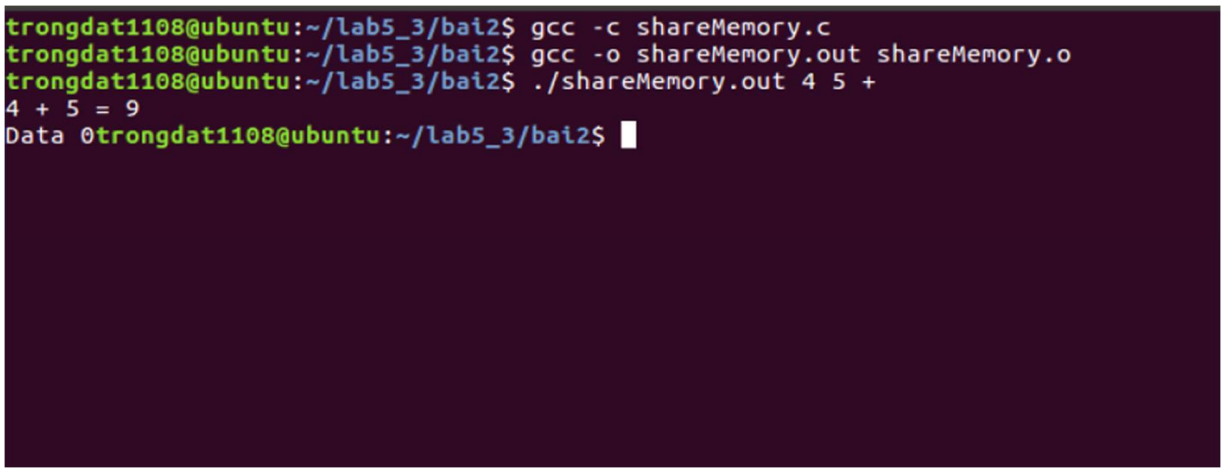
Share Memory:

```
#include <stdio.h>
#include <unistd.h>
#include <limits.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define SIZE 256
int main(int argc, char *argv[])
{
	int *shm, shmid, k, pid;
	key_t key;
	if ((key = ftok(".", 65)) == -1)
	{
		perror("Key created.\n");
		return 1;
	}
	shmid = shmget(key, SIZE, IPC_CREAT | 0666);
	if (shmid == -1)
	{
		perror("Shared memory created.\n");
```

```
        return 2;
}
shm = (int *)shmat(shmid, 0, 0);
pid = fork();
if (pid == 0)
{ // child
        shm[0] = atoi(argv[1]);
        shm[1] = atoi(argv[2]);
        shm[2] = (int)(argv[3][0]);
        sleep(3);
        switch (shm[2])
        {
                case 43:
                        printf("%d+%d=%d\n", shm[0],shm[1],shm[3]);
                        break;
                case 45:
                        printf("%d-%d=%d\n", shm[0],shm[1],shm[3]);
                        break;
                case 120:
                        printf("%d*%d=%d\n", shm[0],shm[1],shm[3]);
                        break;
                case 47:
                        printf("%d/%d=%d\n", shm[0],shm[1],shm[3]);
                        break;
```

```
        }
        shmdt((void *)shm);
        shmctl(shmid, IPC_RMID, (struct shmid_ds *)0);
        return 0;
}
else if (pid > 0)
{ // parent
        printf("Data %d",shm[2]);
        sleep(1);
        if(shm[2]==43){
                shm[3]=shm[1]+shm[0];
        }else if(shm[2]==45){
                shm[3]=shm[1]-shm[0];
        }else if(shm[2]==120){
                shm[3]=shm[1]*shm[0];
        }else if(shm[2]==47){
                shm[3]=shm[0]*1.0/shm[1];
        }
        shmdt((void *)shm);
        sleep(5);
        return 0;
}
else
```

```
        {
                perror("Fork failed.");
                return 4;
        }
        return 0;
}
```



Message queue:

Write.c

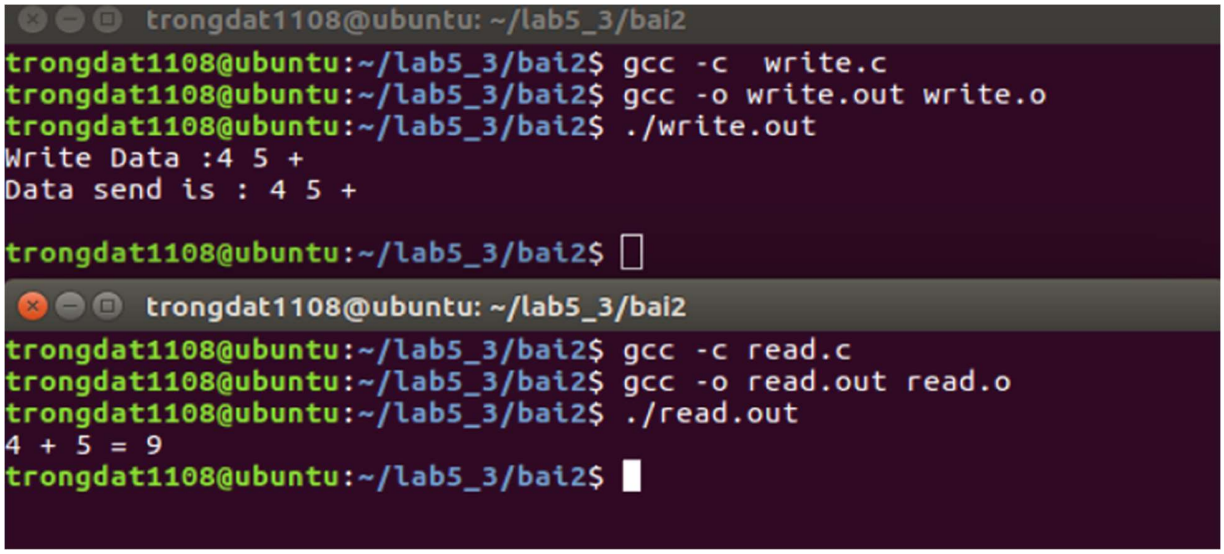
```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>
// structure for message queue
struct mesg_buffer
{
    long mesg_type;
    char mesg_text[100];
} message;
int main()
{
    key_t key;
    int msgid;
    // ftok to generate unique key
    key = ftok("msg.txt", 1);
    // msgget creates a message queue
    // and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);
    message.mesg_type = 1;
    printf("Write Data :");
    fflush(stdin);
    fgets(message.mesg_text, sizeof(message.mesg_text), stdin);
    msgsnd(msgid, &message, sizeof(message), 0);
    // display the message
    printf("Data send is : %s \n", message.mesg_text);
    return 0;
}
```

Read.c

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>
// structure for message queue
struct mesg_buffer
{
    long mesg_type;
    char mesg_text[100];
} message;
int main()
{
    key_t key;
    int msgid;
    // ftok to generate unique key
    key = ftok("msg.txt", 1);
    // msgget creates a message queue
    // and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);
    // msgrcv to receive message
    msgrcv(msgid, &message, sizeof(message), 1, 0);
    // message.mesg_text
    int tmp[20];
    int i, cnt = 0;
    for (i = 0; i < strlen(message.mesg_text); i++)
    {
        if (message.mesg_text[i] != ' ')
        {
            tmp[cnt] = message.mesg_text[i] - '0';
            cnt++;
        }
    }
    tmp[2] += '0';
    switch (tmp[2])
    {
        case 43:
            printf("%d + %d = %d\n", tmp[0], tmp[1], tmp[0] + tmp[1]);
            break;
```

```c
    switch (tmp[2])
    {
        case 43:
            printf("%d + %d = %d\n", tmp[0], tmp[1], tmp[0] + tmp[1]);
            break;
        case 45:
            printf("%d - %d = %d\n", tmp[0], tmp[1], tmp[0] - tmp[1]);
            break;
        case 120:
        case 42:
            printf("%d * %d = %d\n", tmp[0], tmp[1], tmp[0] * tmp[1]);
            break;

        case 47:
            printf("%d / %d = %f\n", tmp[0], tmp[1], tmp[0] * 1.0 / tmp[1]);
            break;
    }
    // to destroy the message queue
    msgctl(msgid, IPC_RMID, NULL);
    return 0;
}
```



Bài thêm:

Share Memory:

```
#include <stdio.h>
#include <unistd.h>
#include <limits.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include<time.h>

#define SIZE 256
int main(int argc, char* argv[])
{
```

```
srand(time(NULL));
int i;

int *shm, shmid, k,pid;
key_t key;
if((key=ftok(".",65))==-1){
        perror("Key created.\n");
        return 1;
}
shmid = shmget(key, SIZE, IPC_CREAT | 0666);
if (shmid == -1) {
        perror("Shared memory created.\n");
        return 2;
}
shm = (int*) shmat(shmid, 0, 0);
pid = fork();
if(pid==0) { // child
        FILE *f = fopen("data", "w");
        int n = atoi(argv[1]);
        for (i = 0; i < n; ++i)
        {
                fprintf(f,"%d\n", rand() % 100);
        }
        fclose(f);
```

```
        FILE *f1 = fopen("data","r");
        int k,x=1;
        shm[0]=n;
        shm[shm[0]+1]=-1;
        while(fscanf(f1,"%d",&k) != EOF)
        {
                shm[x] = k;
                x++;
        }
        fclose(f1);
        //
        sleep(3);
        printf("Sum=%d\n",shm[shm[0]+1]);
        printf("Mang sau khi sap xep:\n");
        for (i = 1; i <= shm[0]; ++i)
        {
                printf("%d ",shm[i]);
        }
        shmdt((void*) shm);
        shmctl(shmid, IPC_RMID, (struct shmid_ds*) 0);
        return 0;
}
else if(pid >0) { // parent
```

```
sleep(1);
int sum=0;
for (i = 1; i <= shm[0]; ++i)
{
        sum+=shm[i];
}
shm[shm[0]+1]=sum;
//sort
int j,k;
for (i = 1; i < shm[0]; ++i)
{
        for (j = 1; j < shm[0]; ++j)
        {
                if (shm[i] < shm[j])
                {
                        k = shm[i];
                        shm[i] = shm[j];
                        shm[j] = k;
                }
        }
}
//TODO
shmdt((void*) shm);
sleep(5);
```

```
            return 0;
        }
        else { perror("Fork failed."); return 4; }
        return 0;
}
```

```
trongdat1108@ubuntu:~/lab5_3/bai3$ gcc -c shareMemory.c
trongdat1108@ubuntu:~/lab5_3/bai3$ gcc -o shareMemory.out shareMemory.o
trongdat1108@ubuntu:~/lab5_3/bai3$ ./shareMemory.out 100
Sum=4994
Mang sau khi sap xep:
0 4 4 6 6 7 8 8 10 12 15 16 16 17 17 17 20 20 21 21 22 23 23 23 23 23 25 27 29 3
1 33 34 37 38 41 41 42 43 43 44 45 45 46 47 48 49 50 50 53 54 54 54 55 55 55 56
56 56 57 57 59 61 62 62 63 65 66 67 68 68 69 69 70 70 71 73 74 75 79 80 81 81 82
 83 83 85 85 85 87 89 90 90 91 91 91 92 95 95 98 17 trongdat1108@ubuntu:~/lab5_3
```

Message queue:

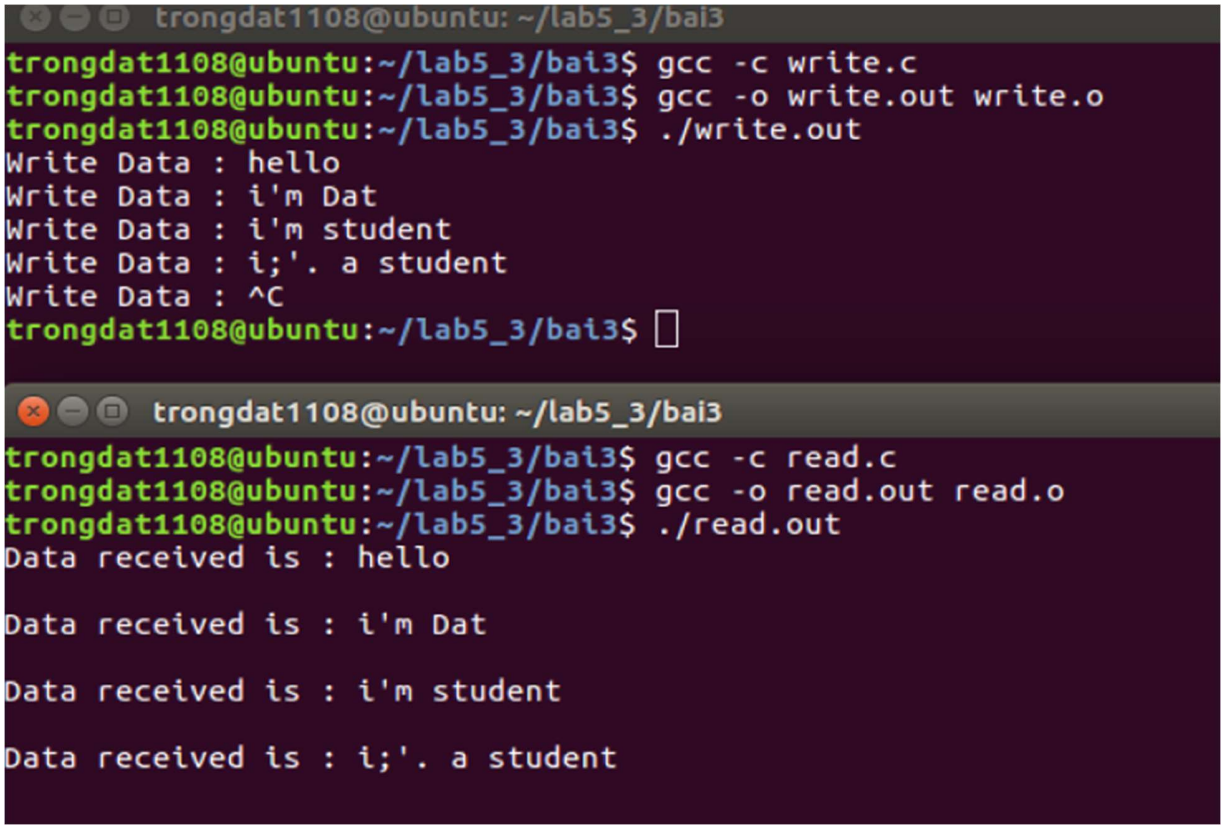
Write.c

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>
// structure for message queue
struct mesg_buffer
{
    long mesg_type;
    char mesg_text[100];
} message;
int main()
{
    key_t key;
    int msgid;
    // ftok to generate unique key
    key = ftok("msg.txt", 1);
    // msgget creates a message queue
    // and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);
    message.mesg_type = 1;
    while(1){
        printf("Write Data : ");
        fgets(message.mesg_text, sizeof(message.mesg_text), stdin);
        msgsnd(msgid, &message, sizeof(message), 0);
        if(strcmp(message.mesg_text, "exit\n") == 0)
            break;
    }
    // msgsnd to send message
    // display the message

    return 0;
}
```

Read.c

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>
// structure for message queue
struct mesg_buffer {
    long mesg_type;
    char mesg_text[100];
} message;
int main()
{
    key_t key;
    int msgid;
// ftok to generate unique key
    key = ftok("msg.txt",1);
// msgget creates a message queue
// and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);
// msgrcv to receive message
// display the message
    while(1){
        msgrcv(msgid, &message, sizeof(message), 1, 0);
        if(strcmp(message.mesg_text, "exit\n") == 0)
            break;
        printf("Data received is : %s \n", message.mesg_text);
    }
// to destroy the message queue
    msgctl(msgid, IPC_RMID, NULL);
    return 0;
}
```


trongdat1108@ubuntu: ~/lab5_3/bai3
trongdat1108@ubuntu:~/lab5_3/bai3$ gcc -c write.c
trongdat1108@ubuntu:~/lab5_3/bai3$ gcc -o write.out write.o
trongdat1108@ubuntu:~/lab5_3/bai3$ ./write.out
Write Data : hello
Write Data : i'm Dat
Write Data : i'm student
Write Data : i;'. a student
Write Data : ^C
trongdat1108@ubuntu:~/lab5_3/bai3$
trongdat1108@ubuntu: ~/lab5_3/bai3
trongdat1108@ubuntu:~/lab5_3/bai3$ gcc -c read.c
trongdat1108@ubuntu:~/lab5_3/bai3$ gcc -o read.out read.o
trongdat1108@ubuntu:~/lab5_3/bai3$ ./read.out
Data received is : hello
Data received is : i'm Dat
Data received is : i'm student
Data received is : i;'. a student