**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP MÔN HỌC NMHĐH**

**LAB 6\_7**

**Nhóm:** **22** **Tổ: 01**

**Tên : Nguyễn Thị Anh Thư**

**MSSV: 51900564**

**NỘI DUNG BÁO CÁO KẾT QUẢ**

**Ví dụ 1.1:**

**- Tiến trình con đọc dữ liệu từ đối số truyền, ghi vào pipe.**

**- Tiến trình cha đọc từ pipe và xuất ra màn hình.**

**KẾT QUẢ THỰC HIỆN**

**1c.1. Phần source code:**

// lab6vd1\_1.c

#include <stdio.h>

#include <unistd.h>

#include <string.h>

int main(int argc, char\* argv[])

{

char result[100];

int fp[2];

int pid;

if(argc<2) {

printf("Thieu doi so.\n");

return -1;

}

if(pipe(fp)==0) {

pid = fork();

if(pid<0) {

printf("Fork that bai.\n");

return -1;

}

else if(pid==0) {

printf("Data from child: %s\n", argv[1]);

close(fp[0]);

write(fp[1], argv[1], strlen(argv[1]));

}

else {

close(fp[1]);

read(fp[0], result, strlen(argv[1]));

printf("Read from child: %s\n", result);

}

}

else {

printf("Pipe failed\n");

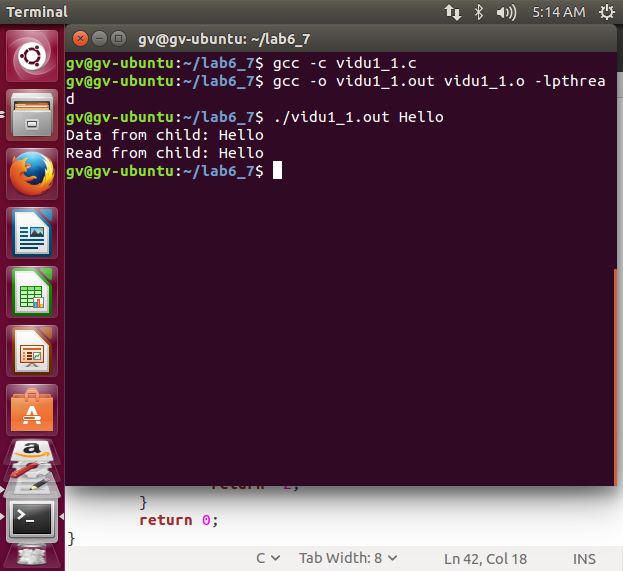
return -2;

}

return 0;

}

**1.1.2. Kết quả chạy chương trình:**

******

**Ví dụ 2.1**: Tiến trình con đọc vào 2 số nguyên từ đối số truyền, ghi vào SM, tiến trình cha thực hiện tính tổng

và ghi lại vào SM. Tiến trình con đọc kết quả và xuất ra màn hình

**KẾT QUẢ THỰC HIỆN**

**2.1.1 Phần source code:**

#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 25

int main(int argc, char\* argv[]) {

int \*shm, shmid, k, pid;

key\_t key;

key=ftok(".",'a');

if(key ==-1) {

perror("Key created.\n");

return -1;

}

shmid = shmget(key, SIZE, IPC\_CREAT | 0666);

if(shmid == -1) {

perror("Shared memory created.\n");

return -1;

}

shm = (int\*) shmat(shmid, NULL, 0);

pid = fork();

if(pid==0) {

shm[0] = atoi(argv[1]);

shm[1] = atoi(argv[2]);

sleep(3);

printf("%d + %d = %d\n", shm[0], shm[1], shm[2]);

shmdt((void\*) shm);

shmctl(shmid, IPC\_RMID, (struct shmid\_ds\*) 0);

return 0;

}

else if(pid >0) { // Tiáº¿n trÃ¬nh cha.

sleep(1);

shm[2] = shm[1] + shm[0];

shmdt((void\*) shm);

sleep(5);

return 0;

}

else {

perror("Fork failed.");

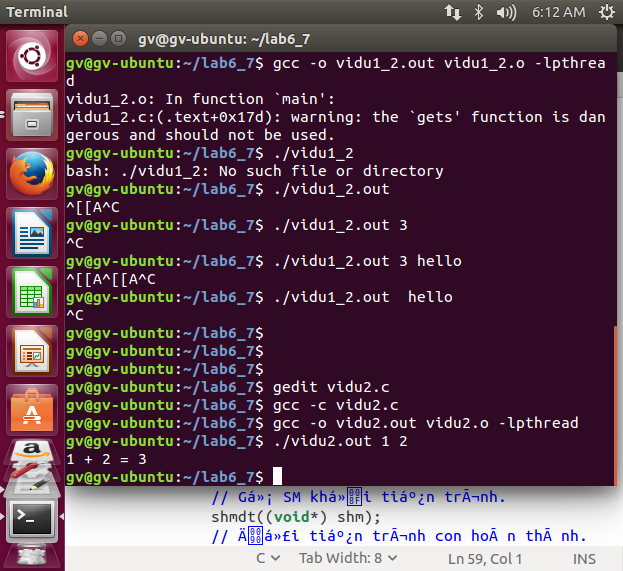
return -1;

}

return 0;

}

**2c.2 Kết quả chạy chương trình:**

****

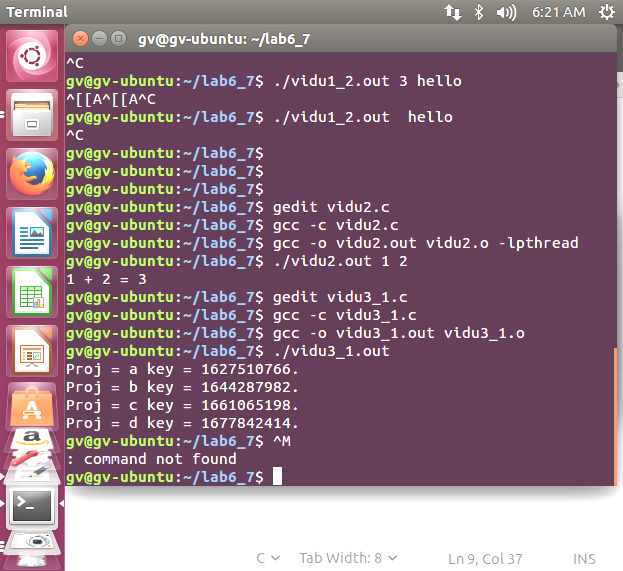
**Vidu3.1**

**KẾT QUẢ THỰC HIỆN**

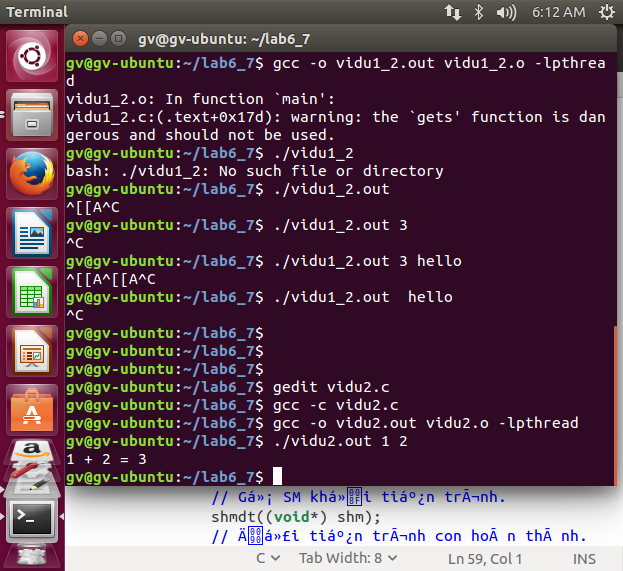
* 1. **.1Phần source code:**

1. #include <stdio.h>
2. #include <sys/types.h>
3. #include <sys/ipc.h>
4. int main(int argc, char\* argv[]) {
5. key\_t key;
6. char i;
7. for(i='a'; i<'e'; i++) {
8. key = ftok(".", i);
9. printf("Proj = %c key = %d.\n", i, key);
10. }
11. return 0;
12. }

**3.1.2Kết quả chạy chương trình:**



**2c.2 Kết quả chạy chương trình:**

****

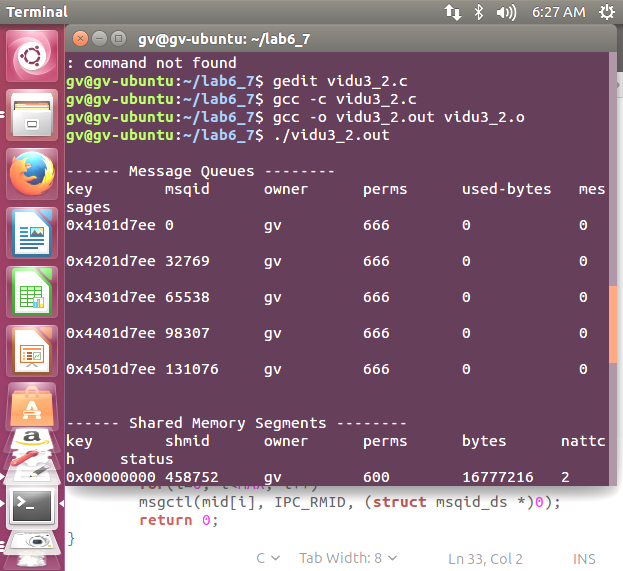
**Vidu3.2**

**KẾT QUẢ THỰC HIỆN**

* 1. **.1Phần source code:**

1. #include <stdio.h>
2. #include <unistd.h>
3. #include <limits.h>
4. #include <sys/types.h>
5. #include <sys/ipc.h>
6. #include <sys/msg.h>
7. #define MAX 5
8. int main(int argc, char\* argv[]) {
9. FILE \*fin;
10. char buffer[PIPE\_BUF], proj='A';
11. int i, n, mid[MAX];
12. key\_t key;
13. for(i=0; i<MAX; i++, proj++) {
14. key = ftok(".", proj);
15. mid[i]=msgget(key, IPC\_CREAT | 0666);
16. if(mid[i]==-1) {
17. perror("Queue created.\n");
18. return 1;
19. }
20. }
21. fin=popen("ipcs","r");
22. n=read(fileno(fin), buffer, PIPE\_BUF);
23. while(n>0) {
24. write(fileno(stdout), buffer, n);
25. n=read(fileno(fin), buffer, PIPE\_BUF);
26. }
27. pclose(fin);
28. for(i=0; i<MAX; i++)
29. msgctl(mid[i], IPC\_RMID, (struct msqid\_ds \*)0);
30. return 0;
31. }

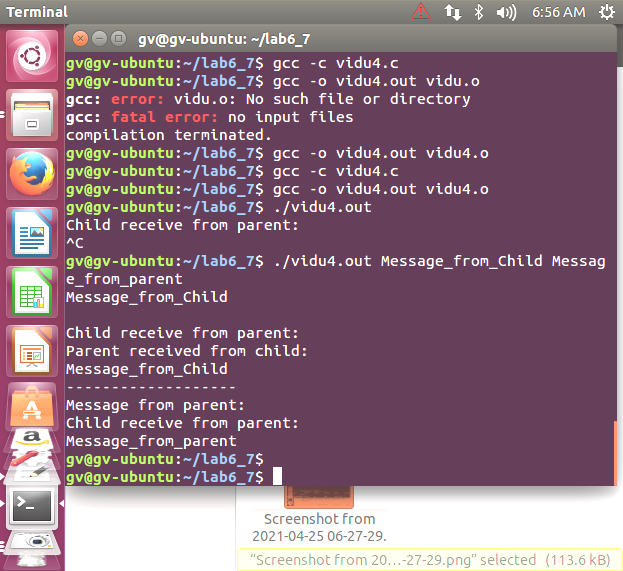
**3.2.2Kết quả chạy chương trình:**

****

**4.1Phần source code:**

1. #include <stdio.h>
2. #include <unistd.h>
3. #include <limits.h>
4. #include <sys/types.h>
5. #include <sys/ipc.h>
6. #include <sys/msg.h>
7. #include <string.h>
8. #include <stdlib.h>
9. struct Message {
10. int mtype;
11. char\* content;
12. };
13. int main(int argc, char\* argv[]) {
14. int mid, id, n, pid, temp;
15. key\_t key;
16. struct Message msg;
17. msg.content=(char\*) malloc(BUFSIZ);
18. key = ftok(".", 'a');
19. if(key==-1) {
20. perror("Key created.\n");
21. return 1;
22. }
23. mid=msgget(key, IPC\_CREAT | 0666);
24. if(mid==-1) {
25. perror("Queue created.\n");
26. return 2;
27. }
28. pid = fork();
29. if(pid==0) {.
30. msg.mtype=10;
31. msg.content=argv[1];
32. n=strlen(msg.content);
33. printf("%s\n", msg.content);
34. n+=sizeof(msg.mtype);
35. temp=msgsnd(mid, &msg, n, 0);
36. if(temp==-1) {
37. perror("Message sent.\n");
38. return 4;
39. }
40. sleep(5);
41. printf("Child receive from parent:\n");
42. n=msgrcv(mid, &msg, BUFSIZ, 11, 0);
43. if(n==-1) {
44. perror("Message received.\n");
45. return 5;
46. }
47. msg.content[strlen(msg.content)]=0;
48. printf("%s\n", msg.content);
49. return 0;
50. }
51. else if(pid >0) {
52. sleep(1);
53. printf("Child receive from parent:\n");
54. n=msgrcv(mid, &msg, BUFSIZ, 10, 0);
55. if(n==-1) {
56. perror("Message received.\n");
57. return 5;
58. }
59. printf("Parent received from child:\n");
60. msg.content[strlen(msg.content)]=0;
61. printf("%s\n", msg.content);
62. printf("-------------------\n");
63. printf("Message from parent:\n");
64. msg.content=argv[2];
65. n=strlen(msg.content);
66. msg.mtype=11;
67. n+=sizeof(msg.mtype);
68. temp=msgsnd(mid, &msg, n, 0);
69. if(temp==-1) {
70. perror("Message sent.\n");
71. return 4;
72. }
73. sleep(10);
74. return 0;
75. }
76. else { perror("Fork failed."); return 3; }
77. return 0;
78. }

**4.2 Kết quả chạy chương trình:**



**KẾT LUẬN**

Qua lab6\_7 em hiểu được cách sử dụng unnamed pipe và named pipe, cách chia sẻ vùng nhớ và hàng đợi message.