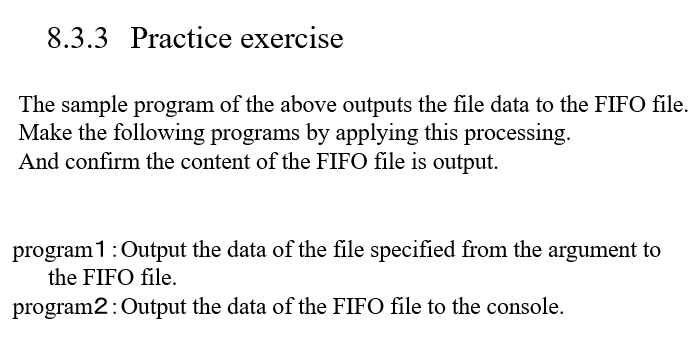
**BT 8.3 8.4 & 8.5**

Nguyễn Hải Dương-20194530

1. Bài tập 8.3:



* Code Program1:

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

    int fd;

    // FIFO file path

    char \* myfifo = "/tmp/myfifo";

    // Creating the named file(FIFO)

    // mkfifo(<pathname>, <permission>)

    mkfifo(myfifo, 0666);

    char arr1[256], dataBuff[256];

    // Open FIFO for write only

    fd = open(myfifo, O\_WRONLY);

    //read dataFile

    FILE \*p = fopen("DataFile.txt", "r");

    fgets(dataBuff, 256, p);

    // Write the input dataBuffing on FIFO

    // and close it

    write(fd, dataBuff, strlen(dataBuff)+1);

    close(fd);

    return 0;

}

* Code program 2:

// C program to implement one side of FIFO

// This side reads first, then reads

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

    int fd1;

    // FIFO file path

    char \* myfifo = "/tmp/myfifo";

    // Creating the named file(FIFO)

    // mkfifo(<pathname>,<permission>)

    mkfifo(myfifo, 0666);

    char str1[256], str2[256];

    // First open in read only and read

    fd1 = open(myfifo,O\_RDONLY);

    read(fd1, str1, 80);

    // Print the read string and close

    printf("Data:\n%s\n", str1);

    close(fd1);

    unlink(myfifo);

    return 0;

}

* Datafile:

NguyenHaiDuong\_20194530. Hello, this is just a random text for testing 8.3.

* Kết quả thực thi chương trình:

A screenshot of a computer

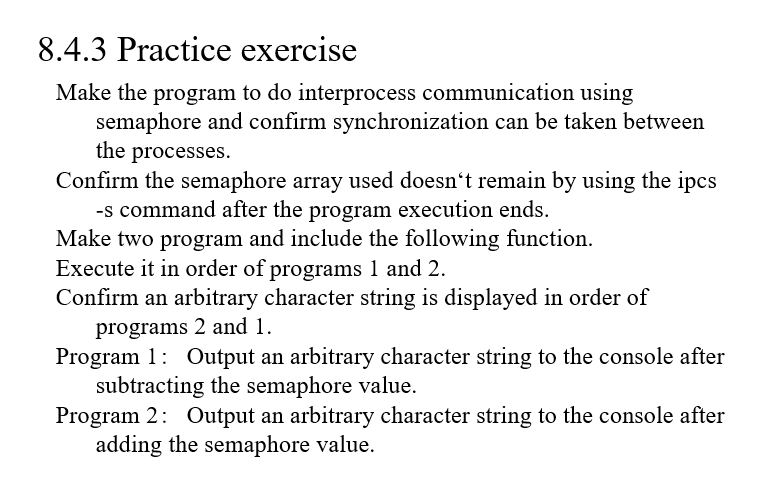
Description automatically generated with medium confidence

* Giải thích:

Program 2 chạy trước sẽ đọc File FIFO và in ra màn hình, tuy nhiên chưa chạy Program 1 nên file FIFO đang rỗng

Khi chạy Program 1 FIFO được ghi dữ liệu của file Data nên program 2 có thể đọc và in ra màn hình như trên.

1. Bài tập 8.4:



* Code:

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#define KEYFILE\_PATH  "/tmp/keypath"

#define ID 'S'

#define SEM\_OK 0

#define SEM\_NG -1

int main(void)

{

  int           semid;

  int           semval;

  key\_t         keyval;

  struct sembuf buff;

  printf("process1       : start\n");

  keyval = ftok(KEYFILE\_PATH, (int)ID);

  // Semaphore ID acquisition

  semid = semget(keyval, 1, IPC\_CREAT|0660);

  printf("processA(sem-) : semID = %d\n", semid);

// Sembuf structure setting for semaphore operation

  buff.sem\_num = (ushort)0;

  buff.sem\_op = (short)-4;

  buff.sem\_flg = (short)0;

  //Semaphore operation

  semop(semid, &buff, 1);

  printf("processA(sem-) : sem\_op = %d\n", buff.sem\_op);

  // Control of semaphore (semaphore value acquisition)

  semval = semctl( semid, 0, GETVAL, 0);

  printf("processA(sem-) : semval = %d\n", semval);

  // Control of semaphore (semaphore value acquisition)

  if(semval == 0){

    // Control of semaphore (semaphore set deletion))

    if(semctl(semid, 0, IPC\_RMID, 0) == SEM\_NG){

      perror("processA       : semctl");

      exit(1);

    }

  }

  printf("processA       : end\n");

  return 0;

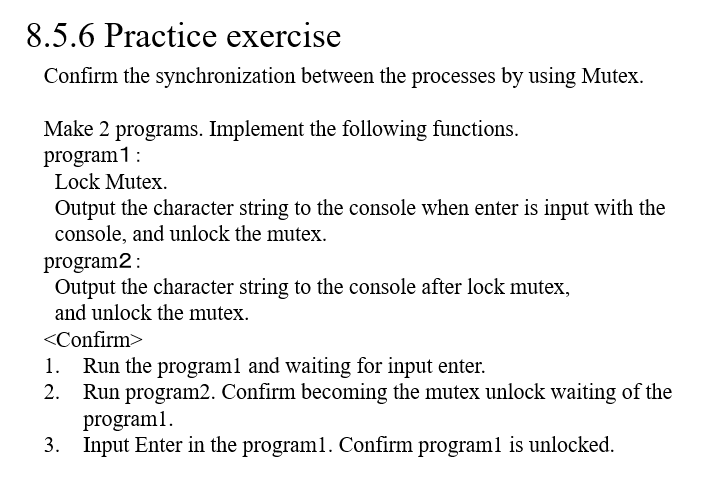
}

* Thực thi chương trình:

A screen shot of a computer

Description automatically generated with medium confidence

1. Bài tập 8.5:



* Code:

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int main()

{

  pthread\_mutex\_t pm;

  pthread\_mutexattr\_t pmattr;

  // Initialize the mutex attribute object

  pthread\_mutexattr\_init(&pmattr);

  // Set mutex attribute(Setting for using mutex to inter process)

  if (pthread\_mutexattr\_setpshared(&pmattr, PTHREAD\_PROCESS\_SHARED) != 0) {

    perror("pthread\_mutexattr\_setpshared");

    exit(1);

  }

 // Initialize mutex

  pthread\_mutex\_init(&pm, &pmattr);

  // Lock mutex

  if (pthread\_mutex\_lock(&pm) != 0) {

    perror("pthread\_mutex\_lock");

    exit(1);

  }

printf("mutex lock\n");

  // Unlock mutex

  if (pthread\_mutex\_unlock(&pm) != 0) {

    perror("pthread\_mutex\_unlock");

    exit(1);

  }

  printf("mutex unlock\n");

  return 0;

}

* Thực thi chương trình:

A screenshot of a computer

Description automatically generated with low confidence