Practice Process, Thread, and IPC Concepts

fork.c

1. Execute the program to understand and answer each question mentioned in the source code file
   a. Get the program back to the original state for each question
   b. Question 1: Which process prints this line? What is printed?
   c. Question 2: What will be printed if this line is commented?
   d. Question 3: When is this line reached/printed?
   e. Question 4: What happens if the parent process is killed first? Uncomment the next two lines.

mfork.c

1. Execute the program once to understand and answer the question
    a. Question 1: How many processes are created? Explain.

pipe-sync.c

1. Update the program to answer the question in the source code file.
    a. Hint: The read and write system call could be useful
    b. Look at the man pages if you don’t know how to use them

fifo_producer.c and fifo_consumer.c

1. Create a fifo and open it for writing and reading, respectively
   a. Templates on following two slides
2. Compile the programs
3. Open 4 terminals and answer the following questions
   a. What happens if you only launch a producer (but no consumer)?
   b. What happens if you only launch a consumer (but no producer)?
   c. If one producer and multiple consumers, then who gets the message sent?
   d. Does the producer continue writing messages into the fifo, if there are no consumers?
   e. What happens to the consumers, if all the producers are killed?

Consumer Example

```
main()
{
      char str[MAX_LENGTH];
      int num, fd;

      mkfifo(FIFO_NAME, 0666); // create FIFO file
      fd = open(FIFO_NAME, O_WRONLY); // open FIFO for writing

      printf("Enter text to write in the FIFO file: ");
      fgets(str, MAX_LENGTH, stdin);
      while(!(feof(stdin))){
            if ((num = write(fd, str, strlen(str))) == -1)
                  perror("write");
            else
                  printf("producer: wrote %d bytes\n", num);
                  fgets(str, MAX_LENGTH, stdin);
      }
}
```

Producer Example

```
main()
{
        char str[MAX_LENGTH];
        int num, fd;

        mkfifo(FIFO_NAME, 0666); // make fifo, if not already present
        fd = open(FIFO_NAME, O_RDONLY); // open fifo for reading

        do{
                if((num = read(fd, str, MAX_LENGTH)) == -1)
                        perror("read");
                else{
                        str[num] = '\0’;
                        printf("consumer: read %d bytes\n", num);
                        printf("%s", str);
                }
        }while(num > 0);
}
```

shared_memory3.c

1. Understand the code
2. Compile/execute the program
3. Question 1: Explain the output

thread-1.c

1. Compile and execute the program
   a. gcc -o thread1 thread-1.c –pthread
   b. ./thread1
2. Observe and execution and answer the two questions referenced in the source code file
   a. Question 1: Are changes made to the local or global variables by the child process reflected in the parent process? Explain.
   b. Question 2: Are changes made to the local or global variables by the child thread reflected in the parent process? Separately explain what happens for the local and global variables.