PROJECT 1 PART 1 Deliverables

Your project deliverables are to be posted to CUNY Blackboard using the Project 1 link.

1) Only one source code file as follows with filename format showing your last name and student id: MYLASTNAME\_MYSTUDENTID.c

a) At top of the file: your full name in comments, for example: // First\_Name Last\_Name

b) In-line code comments no fewer than every 5 lines of code which describe your algorithmic steps in plain English language.

That is, I do not want to see literal syntax transcriptions of the code.

2) An MS Word or PDF document with a write-up of your software design and an explanation of the issues encountered and resolved. (1 page)

The write-up is essay style with no bullet points nor itemized lists nor source code nor pseudo-code listings.

Filename format is MYLASTNAME\_MYSTUDENTID.doc (or pdf.

3) Text file of standard output to screen according to output format in (I) of Project 1 Part 1 Description and Requirements (below)

for your program execution with command line arguments CHUNK\_SIZE = 48 and BUFFER\_SIZE = 10 and the source input file that I have

included in CUNY Blackboard for the project.

Project 1 Part 1 Description and Requirements

--------------------------------------------

(A) In your first project you will create a program that, when executed, has both a parent process and a child process. You use the fork()system call to create a child process as given in the lectures for Chapter 3 Process Creation and Termination. Please refer to your lecture notes.

(B) The parent process code branch utilizes the Producer Algorithm for Shared Memory while the child process code branch makes use of

the Consumer Algorithm for Shared Memory as given in the lectures for Chapter 3 Producer and Consumer Model for Shared Memory.

Please refer to your lecture notes. The BUFFER\_SIZE is to be accepted by your program as a command line argument.

(C) The Parent Producer and Child Consumer Algorithms are to be implemented using the POSIX Shared Memory API as given in the lectures

for Chapter 3 POSIX Shared Memory API.

(D) The Parent Process produces the file contents into shared memory as read from a source file (SRC\_FILE) in chunks using read() (as was discussed in the Chapter 3 lectures) while the Child Process writes the shared memory items to a target file (TGT\_FILE) using write() (as was discussed in the Chapter 3 lectures). The chunk size (CHUNK\_SIZE) is the same for both read() and write(). It will be accepted as input by your program as a command line argument.

(E) You are responsible for designing and implementing a strategy for the termination of the Parent and Child processes when the entire source file has been copied to the the target file.

(F) You may use the C standard library malloc() function for dynamic memory allocation. Otherwise, points will be deducted for not using

the system calls elsewhere.

(G) I will compile and link your source code to generate an executable file using the following command. Therefore, you have to verify that your program can be compiled and linked using the GNU C compiler as was discussed in lectures:

gcc MYLASTNAME\_MYSTUDENTID.c -o MYLASTNAME\_MYSTUDENTID.exe

(H) I will run your program as follows with different values of command line arguments in the following order:

./MYLASTNAME\_MYSTUDENTID.exe SRC\_FILE TGT\_FILE CHUNK\_SIZE BUFFER\_SIZE

(I) Your program will generate output to the screen from the child (consumer) process for each item consumed from shared memory in the

following format using write():

"CHILD: OUT = %d\n"

"CHILD: ITEM = %s%\n"

Your program will also generate output to the screen from the parent (producer) process for each item produced to shared memory in

the following format using write():

"PARENT: IN = %d\n"

"PARENT: ITEM = %s%\n"