Software Requirements Specification

for

Assignment 1

Version 1.0

Prepared by Adam Laviguer and Hammad Qureshi

CPSC 351 Section 03

Created on March 3, 2020

Table of Contents

Revision History

Name	Date	Description of Changes	Version
Adam Laviguer	3/03/2020	Initial document creation	Draft
Adam Laviguer	3/26/2020	Document finalization	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to describe the implementation of an application which synchronously transfers files between two processes called ***sender*** and ***receiver***.

2. Overall Description

Use shared memory and message queues to transfer data between two processes.

2.1 Program Features

sender.cpp

The process that sends files to the receiver process. It is invoked as `$ ./sender <file.txt>` and it performs the following steps:

1. Attach to the shared memory segment and connect to the message queue, both of which were setup by the `receiver` process.
2. Read a predefined number of bytes from the specified file and store in shared memory.
3. Send a message to the receiver that contains a `size` field to indicate how many bytes were read from the file.
4. Wait on the message queue to receive a confirmation from the `receiver` process.
5. Repeat from Step 2 until the whole file has been read.
6. When the end of the file is reached, send a message to the `receiver` process with the `size` field set to 0.
7. Close the file, detach from shared memory, and exit.

recv.cpp

The process that receives files from the sender process. It is invoked as `$ ./receiver` and it performs the following steps:

1. Attach to the shared memory segment.
2. Connect to the message queue.
3. Wait on the message queue to receive a message from the `sender` process.
4. If the `size` field of the message is not `0`, then the `receiver` reads the `size` field and saves the number of bytes to a file called `recvfile`.
5. Send a message to the sender process acknowledging successful reception and saving of data.
6. Repeat from Step 3 until the `size` field of the message is `0`.
7. Close the file, detach from shared memory, deallocate shared memory and message queues, and exit.

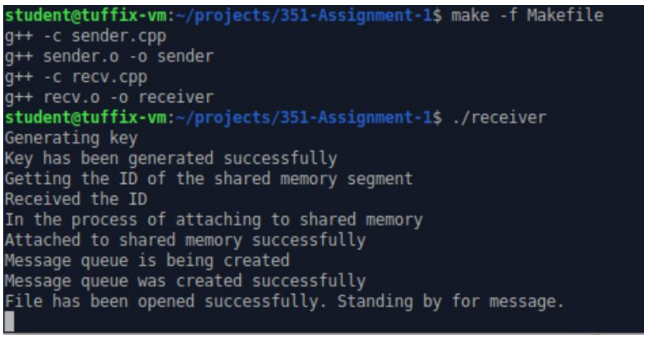
2.2 Operating Environment

The application officially operates in Tuffix OS, a distribution of Linux OS.

2.3 User Documentation

To run the application, perform the following steps:

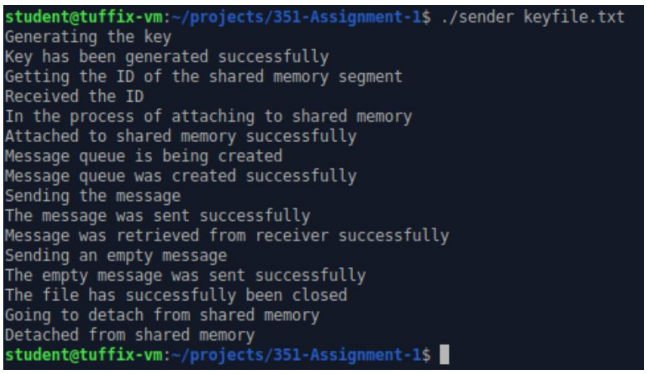
1. Download the .tar file to your desired location and extract
2. Open a terminal window and navigate to the directory containing the extracted application
3. Run the `Makefile` by entering `$ make -f Makefile`
4. Run the `sender` process by entering `$ ./sender keyfile.txt`
5. Open a new terminal window and navigate to the directory containing the extracted application
6. Run the `receiver` process by entering `$ ./receiver`



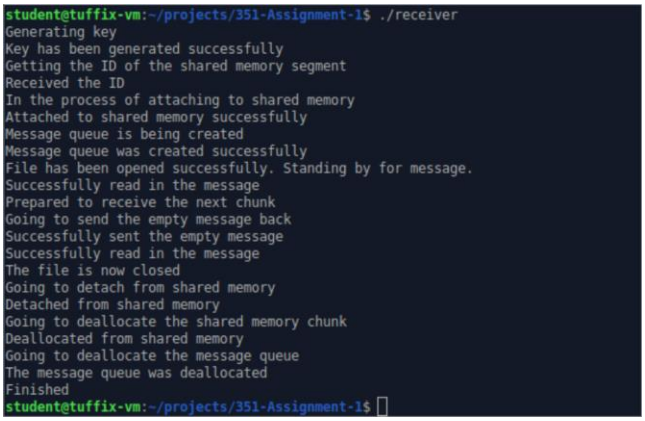
Terminal 1

Note

Steps 4 and 6 may be interchanged and the functionality of the application will not be affected, as shown in the screenshots to the left.



Terminal 2

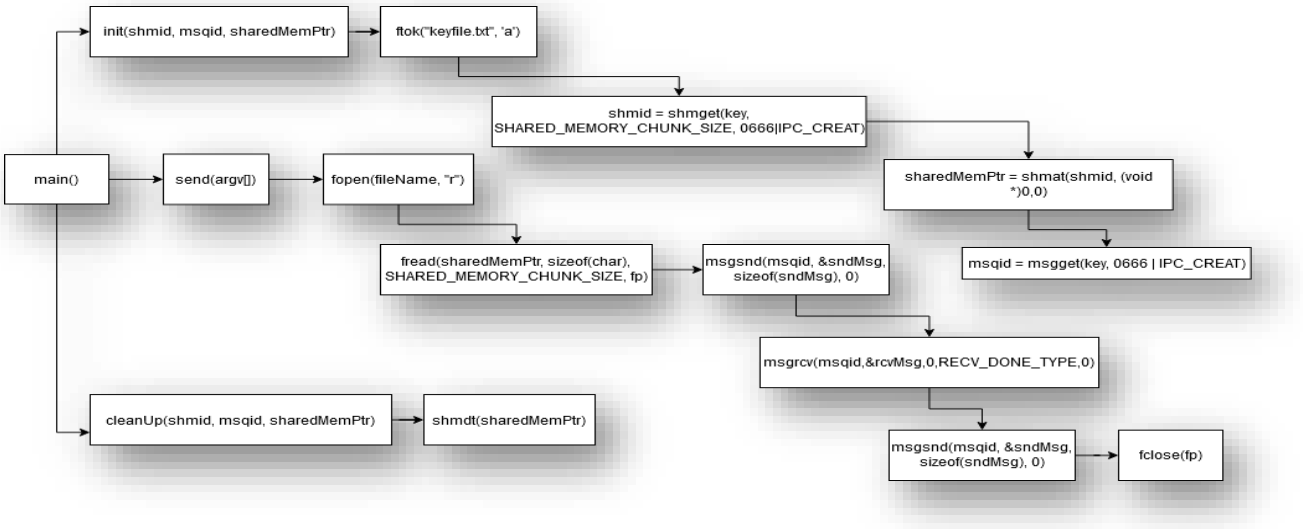


Terminal 1 – End of Program

3. System Design

3.1 Data Flow Diagrams

sender.cpp



recv.cpp