OS Lab 4

Session: 2021 – 2025

Submitted by:

Muhammad Yaqoob 2021-CS-118

Supervised by:

Ms. Abqa Javed

Department of Computer Science

University of Engineering and Technology

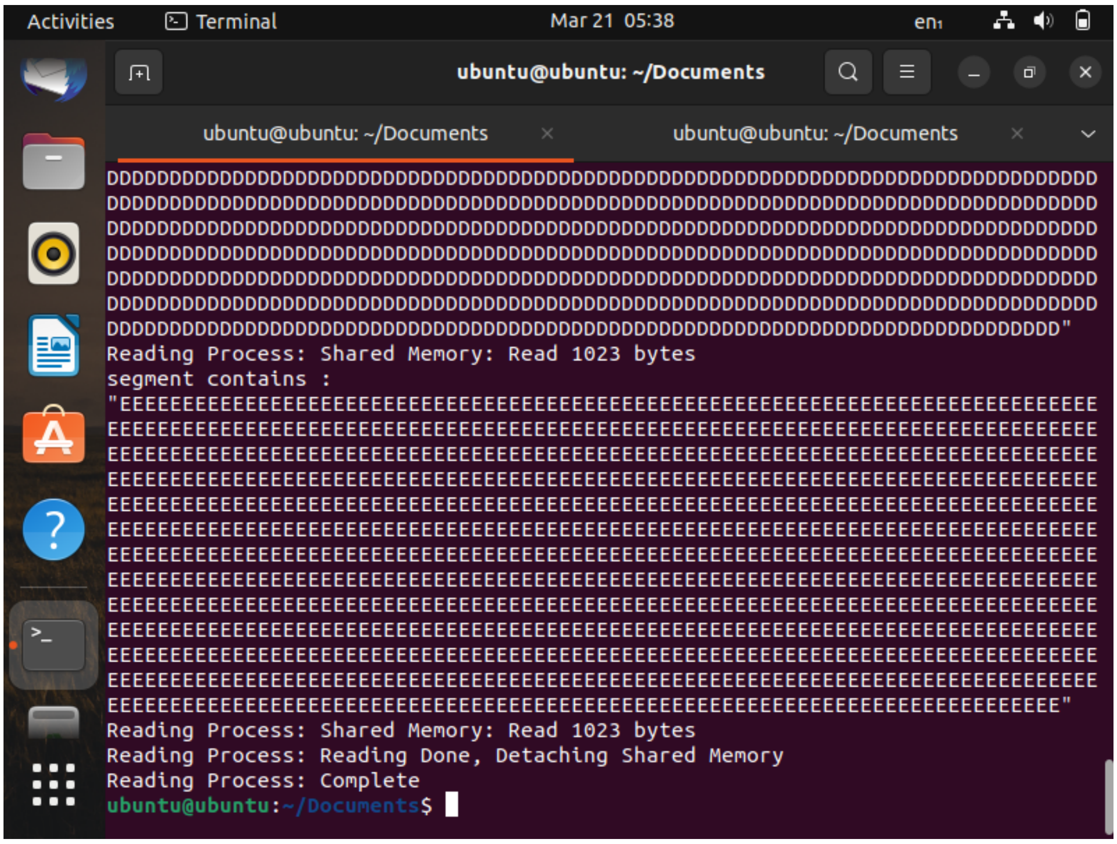
Lahore Pakistan

0.0.1 Question:

Write a C/C++ program to demonstrate inter-process communication using shared memory. Implement the following: Process 1 creates the shared memory and writes some string into shared memory then waits for other process to read and modify the contents of shared memory.
Process 2 gets the shared memory created by process 1, read the string, display the actual string and reverse of the string and modifies the content of shared memory.

0.1 Read

0.1.1 Screenshot


Activities
Terminal
Mar 21 05:38
en
ubuntu@ubuntu: ~/Documents
ubuntu@ubuntu: ~/Documents
ubuntu@ubuntu: ~/Documents

```
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD"
Reading Process: Shared Memory: Read 1023 bytes
segment contains :
"EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE"
Reading Process: Shared Memory: Read 1023 bytes
Reading Process: Reading Done, Detaching Shared Memory
Reading Process: Complete
ubuntu@ubuntu:~/Documents$
```

FIGURE 1: Read

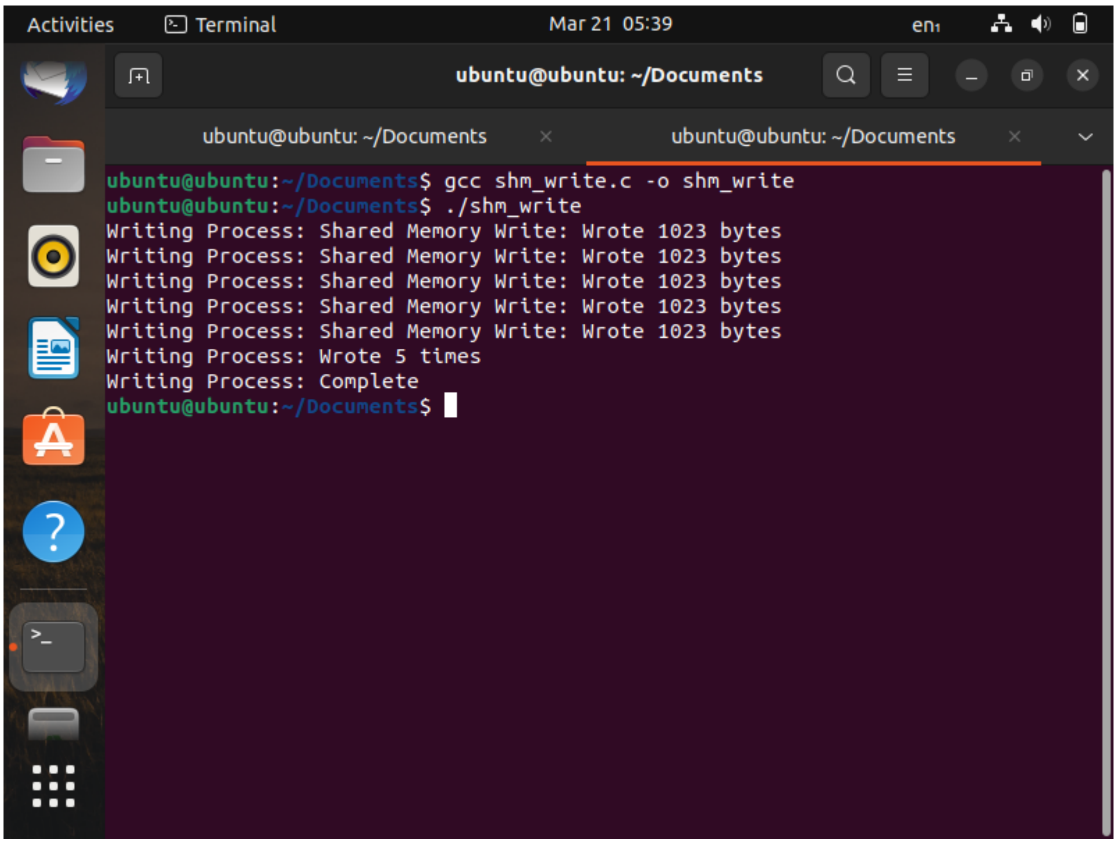
0.1.2 Code:

```
/* Filename: shm_read.c */
#include<stdio.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<sys/types.h>
#include<string.h>
#include<errno.h>
```

```
#include<stdlib.h>

#define BUF_SIZE 1024
#define SHM_KEY 0x1234

struct shmseg {
   int cnt;
   int complete;
   char buf[BUF_SIZE];
};

int main(int argc, char *argv[]) {
   int shmid;
   struct shmseg *shmp;
   shmid = shmget(SHM_KEY, sizeof(struct shmseg), 0644|IPC_CREAT);
   if (shmid == -1) {
      perror("Shared memory");
      return 1;
   }

   // Attach to the segment to get a pointer to it.
   shmp = shmat(shmid, NULL, 0);
   if (shmp == (void *) -1) {
      perror("Shared memory attach");
      return 1;
   }

   /* Transfer blocks of data from shared memory to stdout*/
   while (shmp->complete != 1) {
      printf("segment contains : \n\"%s\"\n", shmp->buf);
      if (shmp->cnt == -1) {
         perror("read");
         return 1;
      }
      printf("Reading Process: Shared Memory: Read %d bytes\n", shmp->cnt);
      sleep(3);
   }
   printf("Reading Process: Reading Done, Detaching Shared Memory\n");
   if (shmdt(shmp) == -1) {
      perror("shmdt");
      return 1;
   }
   printf("Reading Process: Complete\n");
   return 0;
}
```

0.2 Write

0.2.1 ScreenShort


Activities
Terminal
Mar 21 05:39
en₁
ubuntu@ubuntu: ~/Documents
ubuntu@ubuntu: ~/Documents
ubuntu@ubuntu: ~/Documents

```
ubuntu@ubuntu:~/Documents$ gcc shm_write.c -o shm_write
ubuntu@ubuntu:~/Documents$ ./shm_write
Writing Process: Shared Memory Write: Wrote 1023 bytes
Writing Process: Shared Memory Write: Wrote 1023 bytes
Writing Process: Shared Memory Write: Wrote 1023 bytes
Writing Process: Shared Memory Write: Wrote 1023 bytes
Writing Process: Shared Memory Write: Wrote 1023 bytes
Writing Process: Wrote 5 times
Writing Process: Complete
ubuntu@ubuntu:~/Documents$
```

Figure 2: Write

0.2.2 Code:

```
/* Filename: shm_write.c */
#include<stdio.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<sys/types.h>
#include<string.h>
#include<errno.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>

#define BUF_SIZE 1024
#define SHM_KEY 0x1234

struct shmseg {
   int cnt;
   int complete;
   char buf[BUF_SIZE];
};
```

```
int fill_buffer(char * bufptr, int size);

int main(int argc, char *argv[]) {
   int shmid, numtimes;
   struct shmseg *shmp;
   char *bufptr;
   int spaceavailable;
   shmid = shmget(SHM_KEY, sizeof(struct shmseg), 0644|IPC_CREAT);
   if (shmid == -1) {
      perror("Shared memory");
      return 1;
   }

   // Attach to the segment to get a pointer to it.
   shmp = shmat(shmid, NULL, 0);
   if (shmp == (void *) -1) {
      perror("Shared memory attach");
      return 1;
   }

   /* Transfer blocks of data from buffer to shared memory */
   bufptr = shmp->buf;
   spaceavailable = BUF_SIZE;
   for (numtimes = 0; numtimes < 5; numtimes++) {
      shmp->cnt = fill_buffer(bufptr, spaceavailable);
      shmp->complete = 0;
      printf("Writing Process: Shared Memory Write: Wrote %d bytes\n", shmp->cnt)
    ;
      bufptr = shmp->buf;
      spaceavailable = BUF_SIZE;
      sleep(3);
   }
   printf("Writing Process: Wrote %d times\n", numtimes);
   shmp->complete = 1;

   if (shmdt(shmp) == -1) {
      perror("shmdt");
      return 1;
   }

   if (shmctl(shmid, IPC_RMID, 0) == -1) {
      perror("shmctl");
      return 1;
   }
   printf("Writing Process: Complete\n");
   return 0;
}

int fill_buffer(char * bufptr, int size) {
   static char ch = 'A';
   int filled_count;

   //printf("size is %d\n", size);
   memset(bufptr, ch, size - 1);
   bufptr[size-1] = '\0';
   if (ch > 122)
   ch = 65;
   if ( (ch >= 65) && (ch <= 122) ) {
      if ( (ch >= 91) && (ch <= 96) ) {
         ch = 65;
      }
   }
   filled_count = strlen(bufptr);

   //printf("buffer count is: %d\n", filled_count);
   //printf("buffer filled is:%s\n", bufptr);
   ch++;
   return filled_count;
}
```