OPERATING SYSTEMS LAB - PRACTICAL 6 - SHARED MEMORY

Name - Sakshi Soni
Roll No - 13

AIM -

Implement a C program to demonstrate the concept of Shared Memory.

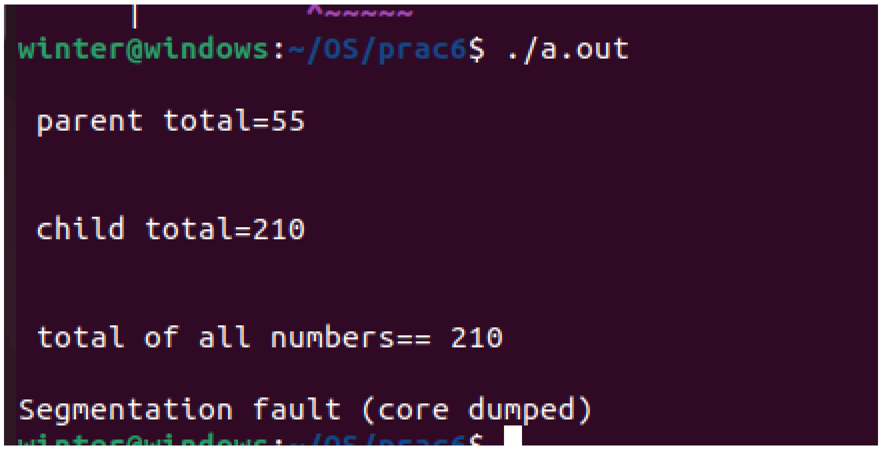
PROGRAM AND OUTPUT -

PROGRAM 1 -
Shared memory basic program to find the total of n numbers.

```
#include<stdio.h>

#include<string.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/shm.h>

#define buf_size 100

int a[]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};

main(void)

{

pid_t pid;

int i;

int *total;
```

```
char b[buf_size];

//get the segment//

int segment_id=shmget(IPC_PRIVATE,2,S_IRUSR|S_IWUSR);

//attach the segment with variable to be used by process

total=(int*)shmat(segment_id,NULL,0);

*total=0;

//creat new child//

pid=fork();

if(pid==0)

{

for(i=10;i<20;i++)

*total= *total + a[i];

sprintf(b,"\n child total=%d \n\n",*total);

write(1,b,strlen(b));

}

else

{

for(i=0;i<10;i++)

*total= *total + a[i];

sprintf(b,"\n parent total=%d \n\n",*total);

write(1,b,strlen(b));

pid=wait(NULL);

if(pid!=-1)
```

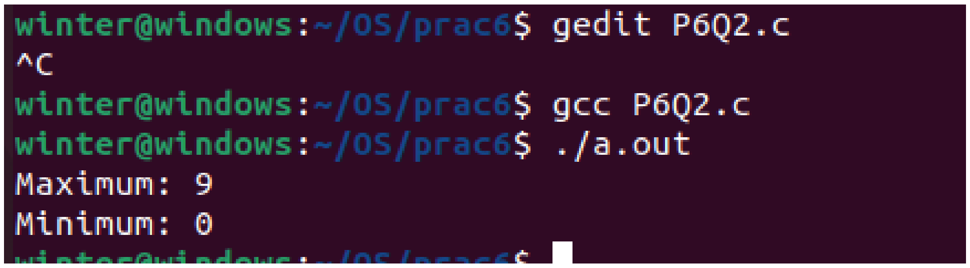
```
        printf("\n total of all numbers== %d\n\n",*total);

        shmdt(total);

        }

        }
```



```
winter@windows:~/OS/prac6$ ./a.out

 parent total=55


 child total=210


 total of all numbers== 210

Segmentation fault (core dumped)
```

PROGRAM 2 -
To find the maximum and minimum element in an array using shared memory.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHM_KEY 12345
#define ARRAY_SIZE 10

typedef struct {
    int max;
    int min;
} SharedData;

int main() {
    int array[ARRAY_SIZE] = {5, 2, 7, 9, 1, 3, 6, 8, 4, 0};

    int shmid = shmget(SHM_KEY, sizeof(SharedData), IPC_CREAT | 0666);
    SharedData *shared_data = (SharedData *)shmat(shmid, NULL, 0);

    shared_data->max = shared_data->min = array[0];
    for (int i = 1; i < ARRAY_SIZE; i++) {
        if (array[i] > shared_data->max) {
            shared_data->max = array[i];
        }
        if (array[i] < shared_data->min) {
            shared_data->min = array[i];
        }
    }

    printf("Maximum: %d\n", shared_data->max);
    printf("Minimum: %d\n", shared_data->min);

    shmdt(shared_data);
    shmctl(shmid, IPC_RMID, NULL);

    return 0;
}
```



```
winter@windows:~/OS/prac6$ gedit P6Q2.c
^C
winter@windows:~/OS/prac6$ gcc P6Q2.c
winter@windows:~/OS/prac6$ ./a.out
Maximum: 9
Minimum: 0
winter@windows:~/OS/prac6$
```

PROGRAM 3 -

Two processes communicating via shared memory: shm_server.c, shm_client.c

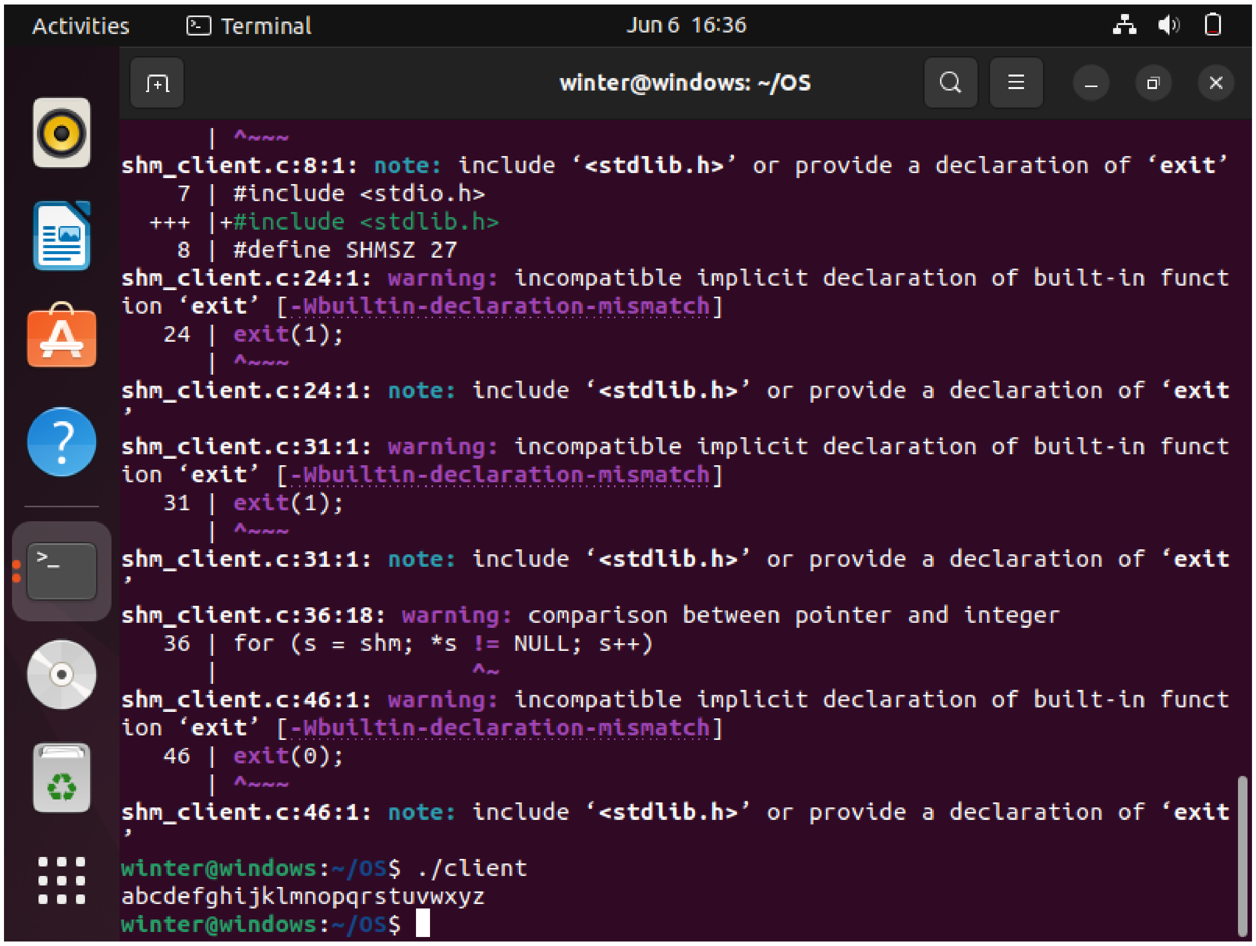
shm_server.c

```
#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#define SHMSZ       27

main()
```

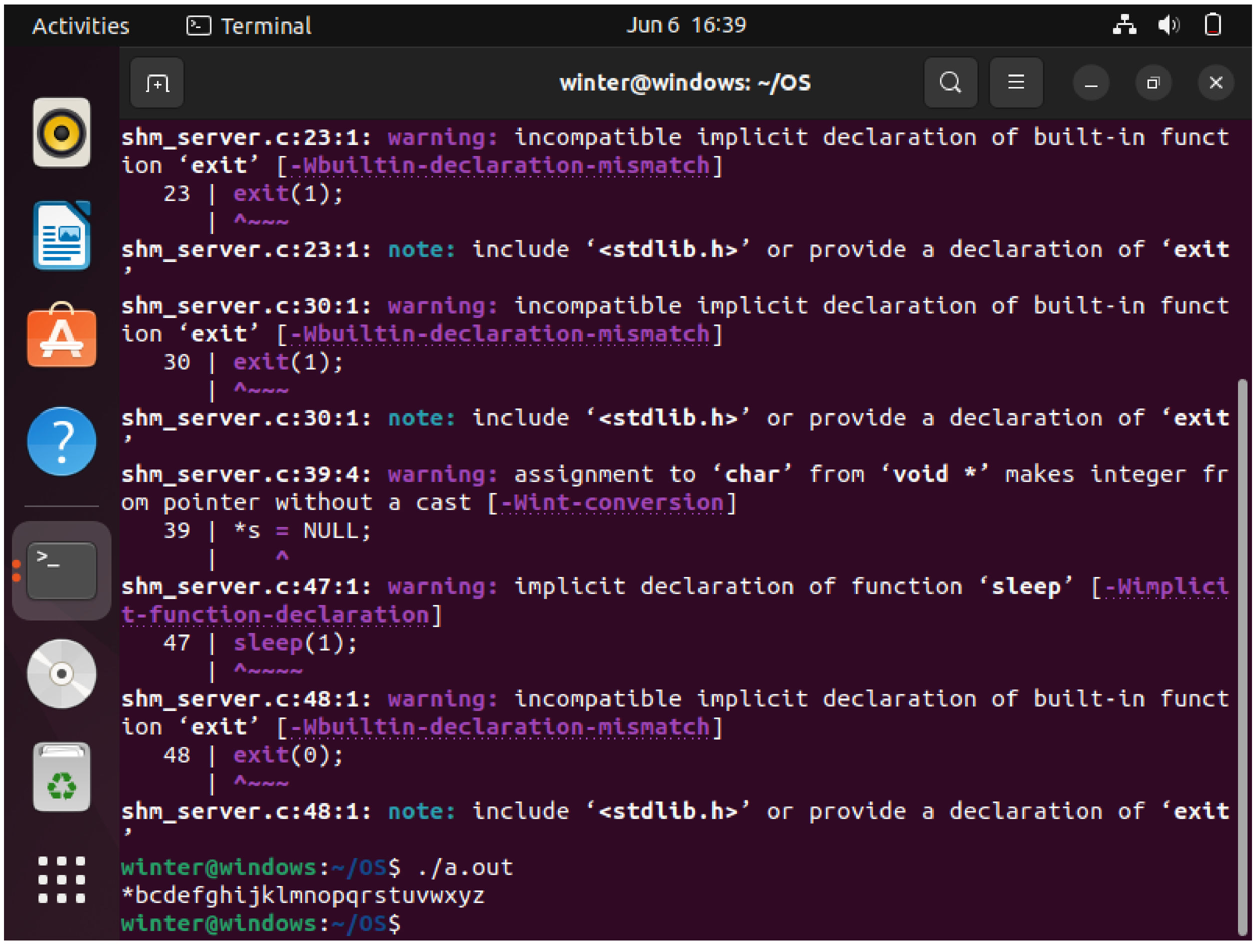
```
{
        char c;
        int shmid;
        key_t key;
        char *shm, *s;
        key = 5678;
        if ((shmid = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0) {
      perror("shmget");
        exit(1);
        }
        if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
        perror("shmat");
        exit(1);
        }
        s = shm;
        for (c = 'a'; c <= 'z'; c++)
        *s++ = c;
        *s = NULL;
 while (*shm != '*')
        sleep(1);
        exit(0);
}
```

shm_client.c

```
#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>


#define SHMSZ       27

 main()

{

        int shmid;

        key_t key;

        char *shm, *s;

        key = 5678;

        if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {

      perror("shmget");

        exit(1);

   }

        if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {

        perror("shmat");

        exit(1);

        }

        for (s = shm; *s != NULL; s++)

        putchar(*s);

        putchar('\n');
        *shm = '*';

        exit(0);

    }
```



```
      | ^~~~
shm_client.c:8:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
    7 | #include <stdio.h>
  +++ |+#include <stdlib.h>
    8 | #define SHMSZ 27
shm_client.c:24:1: warning: incompatible implicit declaration of built-in funct
ion ‘exit’ [-Wbuiltin-declaration-mismatch]
   24 | exit(1);
      | ^~~~
shm_client.c:24:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit
’
shm_client.c:31:1: warning: incompatible implicit declaration of built-in funct
ion ‘exit’ [-Wbuiltin-declaration-mismatch]
   31 | exit(1);
      | ^~~~
shm_client.c:31:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit
’
shm_client.c:36:18: warning: comparison between pointer and integer
   36 | for (s = shm; *s != NULL; s++)
      |                  ^~
shm_client.c:46:1: warning: incompatible implicit declaration of built-in funct
ion ‘exit’ [-Wbuiltin-declaration-mismatch]
   46 | exit(0);
      | ^~~~
shm_client.c:46:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit
’
winter@windows:~/OS$ ./client
abcdefghijklmnopqrstuvwxyz
winter@windows:~/OS$
```



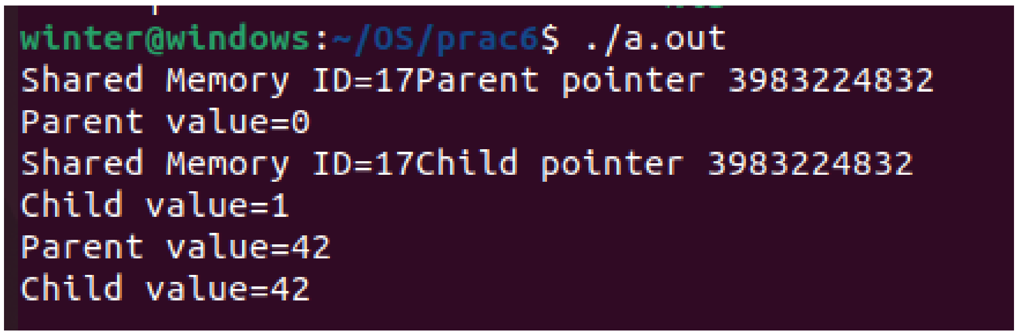
```
shm_server.c:23:1: warning: incompatible implicit declaration of built-in funct
ion ‘exit’ [-Wbuiltin-declaration-mismatch]
   23 | exit(1);
      | ^~~~
shm_server.c:23:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit
’
shm_server.c:30:1: warning: incompatible implicit declaration of built-in funct
ion ‘exit’ [-Wbuiltin-declaration-mismatch]
   30 | exit(1);
      | ^~~~
shm_server.c:30:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit
’
shm_server.c:39:4: warning: assignment to ‘char’ from ‘void *’ makes integer fr
om pointer without a cast [-Wint-conversion]
   39 | *s = NULL;
      |    ^
shm_server.c:47:1: warning: implicit declaration of function ‘sleep’ [-Wimplici
t-function-declaration]
   47 | sleep(1);
      | ^~~~~
shm_server.c:48:1: warning: incompatible implicit declaration of built-in funct
ion ‘exit’ [-Wbuiltin-declaration-mismatch]
   48 | exit(0);
      | ^~~~
shm_server.c:48:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit
’
winter@windows:~/OS$ ./a.out
*bcdefghijklmnopqrstuvwxyz
winter@windows:~/OS$
```

PROGRAM 4 -

Write a C program that illustrates 2 processes communicating using shared memory.

```
#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <unistd.h>

#include <string.h>

#include <errno.h>

int main(void) {
```

```
pid_t pid;

int *shared; /* pointer to the shm */

int shmid;

shmid = shmget(IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666);

printf("Shared Memory ID=%u",shmid);

if (fork() == 0) { /* Child */

/* Attach to shared memory and print the pointer */

shared = shmat(shmid, (void *) 0, 0);

printf("Child pointer %u\n", shared);

*shared=1;

printf("Child value=%d\n", *shared);

sleep(2);

printf("Child value=%d\n", *shared);

} else { /* Parent */

/* Attach to shared memory and print the pointer */

shared = shmat(shmid, (void *) 0, 0);

printf("Parent pointer %u\n", shared);

printf("Parent value=%d\n", *shared);

sleep(1);

*shared=42;

printf("Parent value=%d\n", *shared);

sleep(5);

shmctl(shmid, IPC_RMID, 0);
}

}
```



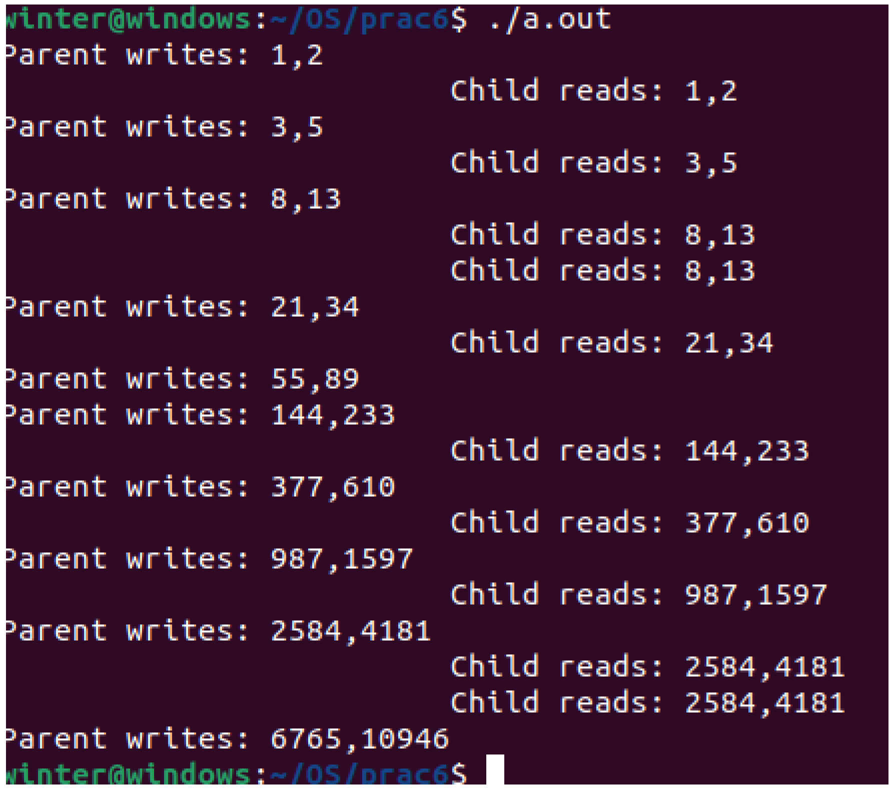
```
winter@windows:~/OS/prac6$ ./a.out
Shared Memory ID=17Parent pointer 3983224832
Parent value=0
Shared Memory ID=17Child pointer 3983224832
Child value=1
Parent value=42
Child value=42
```

<u>PROGRAM 5 -</u>

Sharing Memory between processes

```
#include <stdio.h>

#include <sys/ipc.h>

#include <sys/shm.h>

main()

{

        int shmid. status;

        int *a, *b;

        int i;

        shmid = shmget(IPC_PRIVATE, 2*sizeof(int), 0777|IPC_CREAT);

        if (fork() == 0) {

                        b = (int *) shmat(shmid, 0, 0);

                        for( i=0; i< 10; i++) {

                                sleep(1);

                                printf("\t\t\t Child reads: %d,%d\n",b[0],b[1]);
```

```
                    }

                    shmdt(b);

          }

          else {

                    a = (int *) shmat(shmid, 0, 0);

                    a[0] = 0; a[1] = 1;

                    for( i=0; i< 10; i++) {

                              sleep(1);

                              a[0] = a[0] + a[1];

                              a[1] = a[0] + a[1];

                              printf("Parent writes: %d,%d\n",a[0],a[1]);

                    }

                    wait(&status);

                    shmdt(a);

                    shmctl(shmid, IPC_RMID, 0);

          }

}
```



```
winter@windows:~/OS/prac6$ ./a.out
Parent writes: 1,2
                         Child reads: 1,2
Parent writes: 3,5
                         Child reads: 3,5
Parent writes: 8,13
                         Child reads: 8,13
                         Child reads: 8,13
Parent writes: 21,34
                         Child reads: 21,34
Parent writes: 55,89
Parent writes: 144,233
                         Child reads: 144,233
Parent writes: 377,610
                         Child reads: 377,610
Parent writes: 987,1597
                         Child reads: 987,1597
Parent writes: 2584,4181
                         Child reads: 2584,4181
                         Child reads: 2584,4181
Parent writes: 6765,10946
winter@windows:~/OS/prac6$
```

<u>PROGRAM 6 -</u>
Use of shared memory

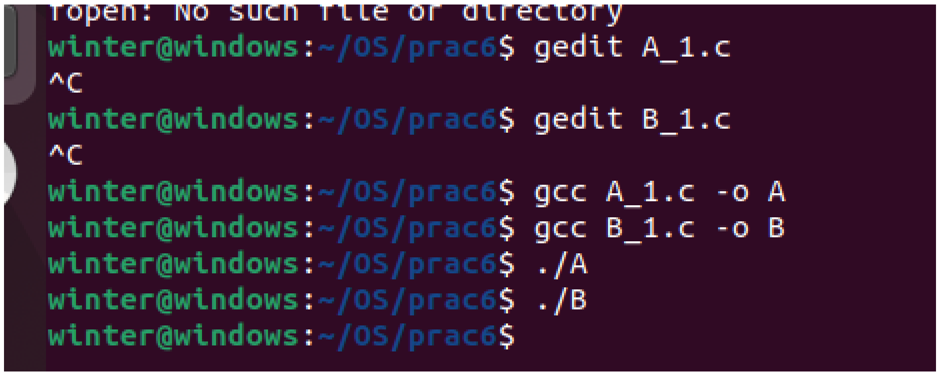
A_1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>

#define MAX_NUMBERS 100

int main() {
   key_t key = ftok("inpfile", 65);  // Unique key for shared memory segment
   int shmid = shmget(key, MAX_NUMBERS * sizeof(int), IPC_CREAT |
0666);
```

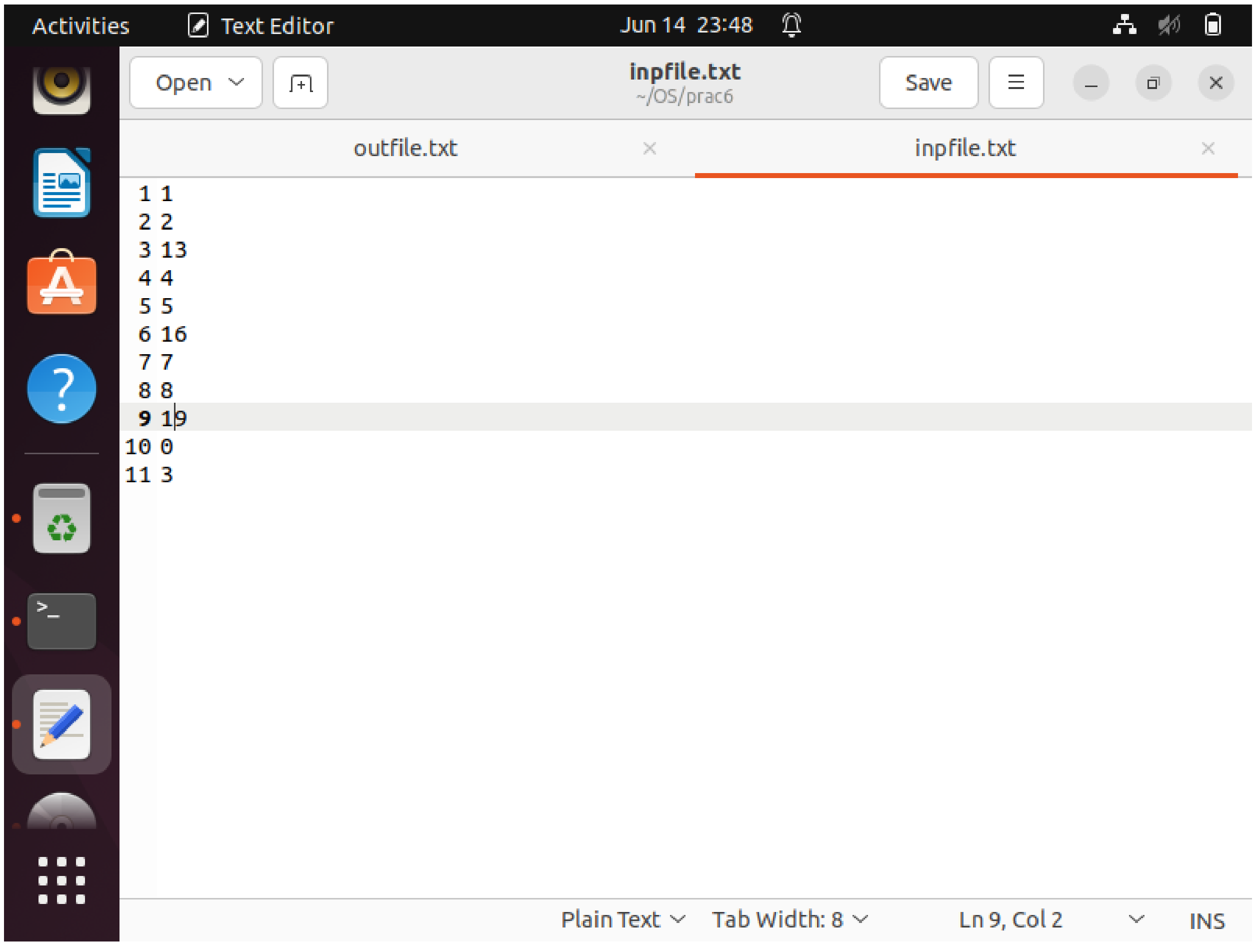
```
    if (shmid == -1) {
        perror("shmget");
        exit(1);
    }

    int* sharedArray = (int*)shmat(shmid, NULL, 0);
    if (sharedArray == (int*)-1) {
        perror("shmat");
        exit(1);
    }

    FILE* file = fopen("inpfile", "r");
    if (file == NULL) {
        perror("fopen");
        exit(1);
    }

    int num;
    int count = 0;

    while (fscanf(file, "%d", &num) == 1 && count < MAX_NUMBERS) {
        sharedArray[count] = num;
        count++;
    }

    fclose(file);

    sleep(5);  // Delay B.c start

    shmdt(sharedArray);  // Detach shared memory segment

    return 0;
}
```

B_1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define MAX_NUMBERS 100

void sortArray(int* array, int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
}

int main() {
    key_t key = ftok("inpfile", 65);  // Same unique key used by A.c
    int shmid = shmget(key, MAX_NUMBERS * sizeof(int), 0666);

    if (shmid == -1) {
        perror("shmget");
        exit(1);
    }

    int* sharedArray = (int*)shmat(shmid, NULL, 0);
    if (sharedArray == (int*)-1) {
        perror("shmat");
        exit(1);
    }

    // Determine the number of integers in the shared array
    int count = 0;
    while (sharedArray[count] != 0 && count < MAX_NUMBERS) {
        count++;
    }

    // Sort the array
    sortArray(sharedArray, count);

    FILE* file = fopen("outfile", "w");
    if (file == NULL) {
        perror("fopen");
        exit(1);
    }

    // Write the sorted array to the file
    for (int i = 0; i < count; i++) {
        fprintf(file, "%d ", sharedArray[i]);
    }

    fclose(file);

    shmdt(sharedArray);          // Detach shared memory segment
    shmctl(shmid, IPC_RMID, 0);  // Delete shared memory segment

    return 0;
}
```



```
fopen: No such file or directory
winter@windows:~/OS/prac6$ gedit A_1.c
^C
winter@windows:~/OS/prac6$ gedit B_1.c
^C
winter@windows:~/OS/prac6$ gcc A_1.c -o A
winter@windows:~/OS/prac6$ gcc B_1.c -o B
winter@windows:~/OS/prac6$ ./A
winter@windows:~/OS/prac6$ ./B
winter@windows:~/OS/prac6$
```



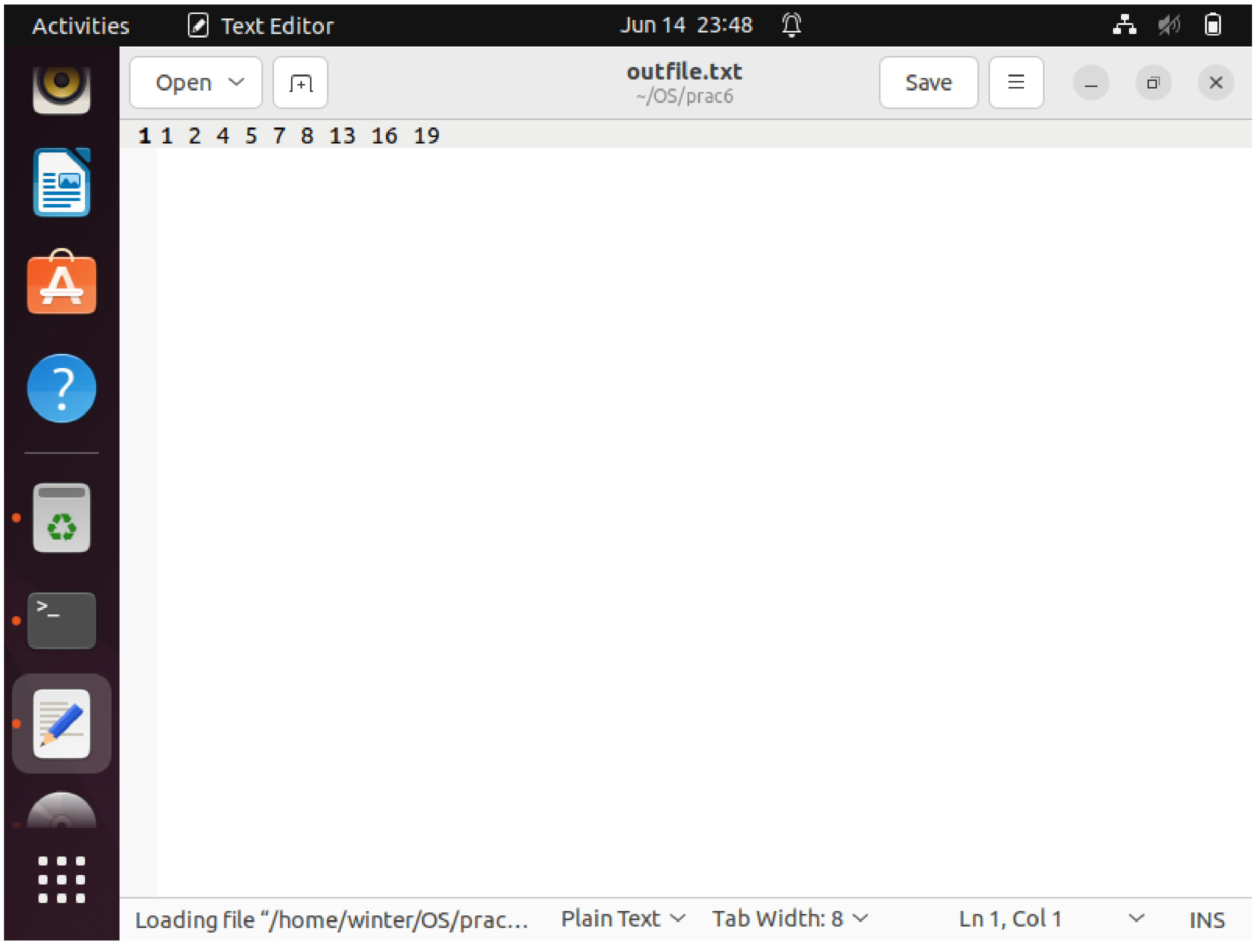
Activities   Text Editor   Jun 14 23:48

inpfile.txt
~/OS/prac6

outfile.txt   inpfile.txt

```
1
2
13
4
5
16
7
8
19
0
3
```

Plain Text   Tab Width: 8   Ln 9, Col 2   INS



```
1 2 4 5 7 8 13 16 19
```

Loading file "/home/winter/OS/prac...

A_2.c

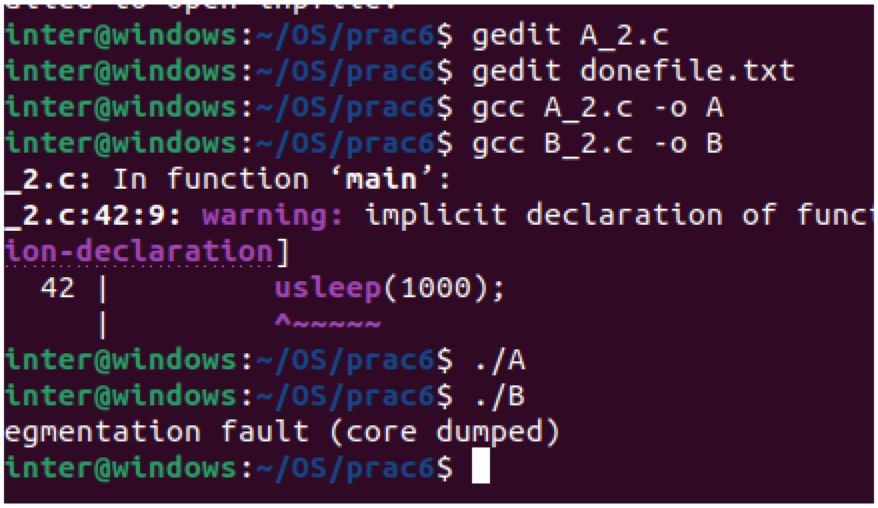
```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/wait.h>
#include <time.h>

#define MAX_INTS 100
```

```
int main() {
    int shmid;
    int *sharedArray;
    int *done;

    // Create shared memory segment for sharedArray
    key_t key_array = ftok("inpfile", 65);
    shmid = shmget(key_array, MAX_INTS * sizeof(int), IPC_CREAT |
S_IRUSR | S_IWUSR);
    sharedArray = (int *)shmat(shmid, NULL, 0);

    // Create shared memory segment for done
    key_t key_done = ftok("donefile", 65);
    int done_shmid = shmget(key_done, sizeof(int), IPC_CREAT | S_IRUSR
| S_IWUSR);
    done = (int *)shmat(done_shmid, NULL, 0);
    *done = 0;

    // Read integers from inpfile and write them to shared array
    FILE *file = fopen("inpfile", "r");
    if (file == NULL) {
        printf("Failed to open inpfile.\n");
        exit(1);
    }

    int numIntegers = 0;
    while (fscanf(file, "%d", &sharedArray[numIntegers]) != EOF) {
        numIntegers++;
        if (numIntegers >= MAX_INTS) {
            break;
        }
    }

    fclose(file);
    // Set done to 1 to indicate A.c has finished writing
    *done = 1;

    // Wait for a few seconds to simulate delay
    sleep(3);

    // Detach and delete shared memory segments
    shmdt(sharedArray);
    shmdt(done);
    shmctl(shmid, IPC_RMID, NULL);
    shmctl(done_shmid, IPC_RMID, NULL);

    return 0;
}
```

B_2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/wait.h>

#define MAX_INTS 100

void bubbleSort(int arr[], int n) {
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
      }
   }
}

int main() {
   int shmid;
   int *sharedArray;
   int *done;

   // Create shared memory segment for sharedArray
   key_t key_array = ftok("inpfile", 65);
   shmid = shmget(key_array, MAX_INTS * sizeof(int), S_IRUSR |
S_IWUSR);
   sharedArray = (int *)shmat(shmid, NULL, 0);

   // Create shared memory segment for done
   key_t key_done = ftok("donefile", 65);
   int done_shmid = shmget(key_done, sizeof(int), S_IRUSR | S_IWUSR);
   done = (int *)shmat(done_shmid, NULL, 0);

   // Wait until done is set to 1 by A.c
   while (*done != 1) {
      // Sleep for a short duration to avoid busy-waiting
      usleep(1000);
   }

   // Determine the number of integers in the shared array
   int numIntegers = 0;
   while (numIntegers < MAX_INTS && sharedArray[numIntegers] != 0) {
      numIntegers++;
   }
```

```
    // Sort the array
    bubbleSort(sharedArray, numIntegers);

    // Write the sorted array to outfile
    FILE *file = fopen("outfile", "w");
    if (file == NULL) {
        printf("Failed to open outfile.\n");
        exit(1);
    }

    for (int i = 0; i < numIntegers; i++) {
        fprintf(file, "%d\n", sharedArray[i]);
    }

    fclose(file);

    // Detach shared memory segments
    shmdt(sharedArray);
    shmdt(done);

    return 0;
}
```



```
inter@windows:~/OS/prac6$ gedit A_2.c
inter@windows:~/OS/prac6$ gedit donefile.txt
inter@windows:~/OS/prac6$ gcc A_2.c -o A
inter@windows:~/OS/prac6$ gcc B_2.c -o B
_2.c: In function ‘main’:
_2.c:42:9: warning: implicit declaration of func
ion-declaration]
  42 |         usleep(1000);
     |         ^~~~~~
inter@windows:~/OS/prac6$ ./A
inter@windows:~/OS/prac6$ ./B
egmentation fault (core dumped)
inter@windows:~/OS/prac6$ 
```

CONCLUSION -

Linux C programs to demonstrate the concept of Shared Memory has been implemented.