**IPC: Shared Memory**

**Subject - Unix Operating System**

**Name – Rachit Bhat**

**PRN – 22610007 Class – TYIT**

**Assignment No – 8b**

**Title-** Write 2 programs. 1st program will take small file from the user and write inside the shared memory. 2nd program will read from the shared memory and write into the file.

**Objectives:**

1. To learn about IPC through message queue.
2. Use of system call and IPC mechanism to write effective application programs.

**Theory:**

Shared Memory is an efficient means of passing data between programs. One program will create a memory portion which other processes (if permitted) can access. Communication between processes using shared memory requires processes to share some variable and it completely depends on how programmer will implement it. One way of communication using shared memory can be imagined like this: Suppose process1 and process2 are executing simultaneously and they share some resources or use some information from other process, process1 generate information about certain computations or resources being used and keeps it as a record in shared memory. When process2 need to use the shared information, it will check in the record stored in shared memory and take note of the information generated by process1 and act accordingly. Processes can use shared memory for extracting information as a record from other process as well as for delivering any specific information to other process. The server maps a shared memory in its address space and also gets access to a synchronization mechanism.

The server obtains exclusive access to the memory using the synchronization mechanism and copies the file to memory. The client maps the shared memory in its address space. Waits until the server releases the exclusive access and uses the data. To use shared memory, we have to perform 2 basic steps:

* Request to the operating system a memory segment that can be shared between processes. The user can create/destroy/open this memory using a shared memory object: An object that represents memory that can be mapped concurrently into the address space of more than one process.
* Associate a part of that memory or the whole memory with the address space of the calling process. The operating system looks for a big enough memory address range in the calling process' address space and marks that address range as a special range.

**Program:**

**Server:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/types.h>

#include <unistd.h>

#define SHM\_SIZE 1024  // Size of shared memory

#define SHM\_KEY 12345  // Key for shared memory

int main() {

    // Prompt the user to provide the input file name

    char input\_filename[256];

    printf("Enter the file name to read: ");

    scanf("%s", input\_filename);

    // Open the input file

    FILE \*file = fopen(input\_filename, "r");

    if (file == NULL) {

        perror("File opening failed");

        exit(1);

    }

    // Read file contents into a buffer

    char file\_contents[SHM\_SIZE];

    size\_t bytes\_read = fread(file\_contents, 1, SHM\_SIZE - 1, file);

    file\_contents[bytes\_read] = '\0';  // Null-terminate the string

    // Close the file after reading

    fclose(file);

    // Create the shared memory segment

    int shmid = shmget(SHM\_KEY, SHM\_SIZE, 0666 | IPC\_CREAT);

    if (shmid == -1) {

        perror("shmget failed");

        exit(1);

    }

    // Attach to the shared memory segment

    char \*shm\_ptr = (char \*)shmat(shmid, NULL, 0);

    if (shm\_ptr == (char \*)-1) {

        perror("shmat failed");

        exit(1);

    }

    // Write the file contents to shared memory

    strcpy(shm\_ptr, file\_contents);

    printf("Writer: File contents written to shared memory.\n");

    // Detach from the shared memory

    shmdt(shm\_ptr);

    return 0;

}

**Client:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/types.h>

#include <unistd.h>

#define SHM\_SIZE 1024  // Size of shared memory

#define SHM\_KEY 12345  // Key for shared memory

int main() {

    // Prompt the user for the output file name

    char output\_filename[256];

    printf("Enter the output file name: ");

    scanf("%s", output\_filename);

    // Create or open the output file for writing

    FILE \*file = fopen(output\_filename, "w");

    if (file == NULL) {

        perror("File opening failed");

        exit(1);

    }

    // Get the shared memory segment

    int shmid = shmget(SHM\_KEY, SHM\_SIZE, 0666);

    if (shmid == -1) {

        perror("shmget failed");

        exit(1);

    }

    // Attach to the shared memory segment

    char \*shm\_ptr = (char \*)shmat(shmid, NULL, 0);

    if (shm\_ptr == (char \*)-1) {

        perror("shmat failed");

        exit(1);

    }

    // Read the data from shared memory and write to the output file

    fprintf(file, "%s", shm\_ptr);

    printf("Reader: Data written to output file.\n");

    // Close the output file

    fclose(file);

    // Detach from the shared memory

    shmdt(shm\_ptr);

    return 0;

}

**Output:**

****

**Conclusion:**

Sharing of files between client and server by using shared memory IPC was implemented.