**EXPERIMENT: 09 Illustrate the concept of inter-process communication using shared memory with a C program.**

**PROGRAM:**

**#include <sys/ipc.h>**

**#include <sys/shm.h>**

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

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <sys/wait.h> // For wait()**

**int main() {**

**// Step 1: Generate a unique key for the shared memory segment.**

**// Here, ftok() is used to create a key based on a file path and an ID.**

**key\_t key = ftok("shmfile", 65); // "shmfile" can be any existing file or a placeholder.**

**// Step 2: Create or get the shared memory segment.**

**// shmget() allocates a shared memory segment of 1024 bytes with read/write permissions.**

**int shmid = shmget(key, 1024, 0666 | IPC\_CREAT);**

**if (shmid == -1) {**

**perror("shmget failed");**

**exit(1);**

**}**

**// Step 3: Fork a child process to demonstrate inter-process communication.**

**pid\_t pid = fork();**

**if (pid < 0) {**

**perror("fork failed");**

**exit(1);**

**}**

**if (pid == 0) { // Child process (reader)**

**// Introduce a delay to ensure the parent has time to write to shared memory.**

**sleep(2);**

**// Step 4: Attach the shared memory segment to the child's address space.**

**char \*shared\_str = (char\*) shmat(shmid, (void\*)0, 0);**

**if (shared\_str == (char\*)-1) {**

**perror("shmat failed in child");**

**exit(1);**

**}**

**// Step 5: Read from the shared memory.**

**printf("Child process read from shared memory: %s\n", shared\_str);**

**// Step 6: Detach the shared memory segment from the child's address space.**

**if (shmdt(shared\_str) == -1) {**

**perror("shmdt failed in child");**

**exit(1);**

**}**

**} else { // Parent process (writer)**

**// Step 4: Attach the shared memory segment to the parent's address space.**

**char \*shared\_str = (char\*) shmat(shmid, (void\*)0, 0);**

**if (shared\_str == (char\*)-1) {**

**perror("shmat failed in parent");**

**exit(1);**

**}**

**// Step 5: Write to the shared memory.**

**const char \*message = "Hello from parent! This is shared memory IPC.";**

**strcpy(shared\_str, message);**

**printf("Parent process wrote to shared memory: %s\n", message);**

**// Step 6: Detach the shared memory segment from the parent's address space.**

**if (shmdt(shared\_str) == -1) {**

**perror("shmdt failed in parent");**

**exit(1);**

**}**

**// Wait for the child process to finish.**

**wait(NULL);**

**// Step 7: Remove the shared memory segment (cleanup).**

**if (shmctl(shmid, IPC\_RMID, NULL) == -1) {**

**perror("shmctl failed");**

**exit(1);**

**}**

**}**

**return 0;**

**}**

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**