Experiment 8

Q1Establish Interprocess communication (IPC) using named pipe.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

int main(){

	int pid, fd1, fd2;
	char buffer[20];

	mkfifo("my_Pipe", 0666);

	pid=fork();
	if(pid > 0){
		//Parent Section
		fd1=open("my_Pipe", O_WRONLY);
		write(fd1, "Hello Child Process\n",20);
	}
	if(pid==0){
		//Child section
		fd2=open("my_Pipe", O_RDONLY);
		read(fd2, buffer, 20);
		printf("%s", buffer);
	}
```

```
        return 0;
}
```

Output:

```
┌──(medhansh㉿Medhansh)-[~]
└─$ gcc namedpipe.c

┌──(medhansh㉿Medhansh)-[~]
└─$ ./a.out
Hello Child Process

┌──(medhansh㉿Medhansh)-[~]
└─$ |
```

Code:

```
  GNU nano 7.2
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

int main(){

        int pid, fd1, fd2;
        char buffer[20];

        mkfifo("my_Pipe", 0666);

        pid=fork();
        if(pid > 0){
                //Parent Section
                fd1=open("my_Pipe", O_WRONLY);
                write(fd1, "Hello Child Process\n",20);
        }
        if(pid==0){
                //Child section
                fd2=open("my_Pipe", O_RDONLY);
                read(fd2, buffer, 20);
                printf("%s", buffer);
        }

        return 0;
}
```

Q2.Establish Interprocess communication (IPC) using message passing technique.

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <unistd.h>
```

```
struct message {
    long msg_type;
    char msg_text[100];
};

int main() {
    key_t key;
    int msgid;
    struct message msg;

    key = ftok("/tmp", 'a');

    // Create a message queue
    msgid = msgget(key, 0666 | IPC_CREAT);
    if (msgid == -1) {
        perror("msgget");
        exit(EXIT_FAILURE);
    }

    // Create a child process
    pid_t child_pid = fork();

    if (child_pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (child_pid == 0) {
        // Child process
        printf("Child process is waiting for a message...\n");
```

```
        msgrcv(msgid, &msg, sizeof(msg), 1, 0);
        printf("Child received: %s", msg.msg_text);
    } else {
        // Parent process
        printf("Parent process is sending a message...\n");

        msg.msg_type = 1;
        strcpy(msg.msg_text, "Hello from the parent!");
        msgsnd(msgid, &msg, sizeof(msg), 0);
    }

    msgctl(msgid, IPC_RMID, NULL);

    return 0;
}
```

Output:

```
┌──(medhansh㉿Medhansh)-[~]
└─$ gcc message.c

┌──(medhansh㉿Medhansh)-[~]
└─$ ./a.out
Parent process is sending a message...
Child process is waiting for a message...
Child received:
┌──(medhansh㉿Medhansh)-[~]
└─$ |
```

```
  GNU nano 7.2
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>
struct message {
    long msg_type;
    char msg_text[100];
};

int main() {
    key_t key;
    int msgid;
    struct message msg;
    key = ftok("/tmp", 'a');
    msgid = msgget(key, 0666 | IPC_CREAT);
    if (msgid == -1) {
        perror("msgget");
        exit(EXIT_FAILURE);
    }
    pid_t child_pid = fork();

    if (child_pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (child_pid == 0) {
        printf("Child process is waiting for a message...\n");
        msgrcv(msgid, &msg, sizeof(msg), 1, 0);
        printf("Child received: %s", msg.msg_text);
    } else {
        printf("Parent process is sending a message...\n");
        msg.msg_type = 1;
        strcpy(msg.msg_text, "Hello from the parent!");
```

```
    if (child_pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (child_pid == 0) {
        printf("Child process is waiting for a message...\n");
        msgrcv(msgid, &msg, sizeof(msg), 1, 0);
        printf("Child received: %s", msg.msg_text);
    } else {
        printf("Parent process is sending a message...\n");
        msg.msg_type = 1;
        strcpy(msg.msg_text, "Hello from the parent!");
        msgsnd(msgid, &msg, sizeof(msg), 0);
    }
    msgctl(msgid, IPC_RMID, NULL);

    return 0;
}
```

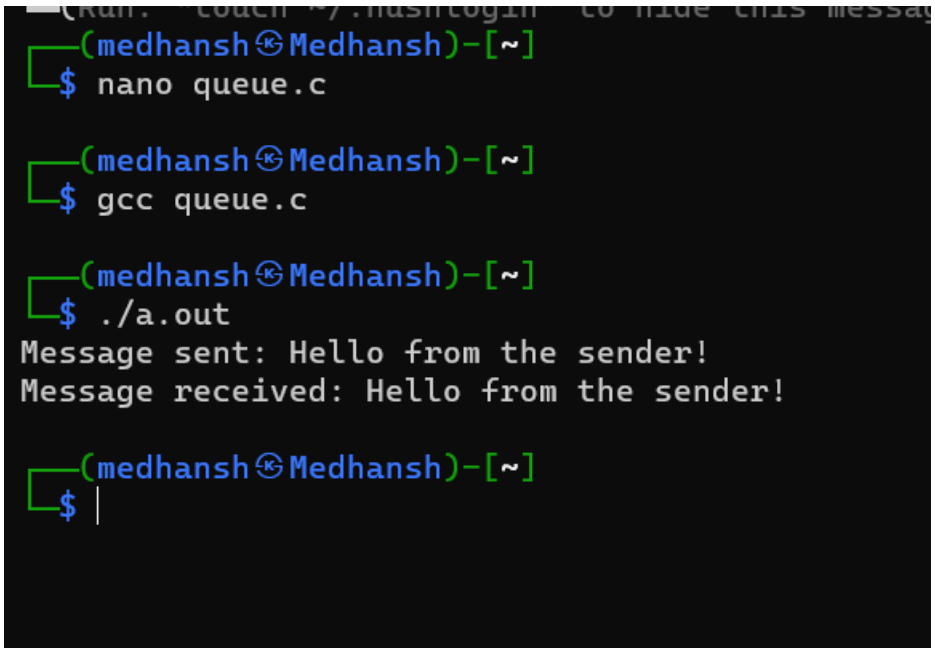
Q3.Establish Interprocess communication (IPC) Message queue Technique.

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>

struct message {
    long msg_type;
    char msg_text[100];
};

int main() {
    key_t key;
    int msgid;
    struct message msg;

    key = ftok("/tmp", 'a');
    msgid = msgget(key, 0666 | IPC_CREAT);

    strcpy(msg.msg_text, "Hello from the sender!");
    msg.msg_type = 1;

    if (msgsnd(msgid, &msg, sizeof(msg.msg_text), 0) == -1) {
        perror("msgsnd");
        exit(EXIT_FAILURE);
    }
```

```
    printf("Message sent: %s\n", msg.msg_text);

    if (msgrcv(msgid, &msg, sizeof(msg.msg_text), 1, 0) == -1) {
        perror("msgrcv");
        exit(EXIT_FAILURE);
    }

    printf("Message received: %s\n", msg.msg_text);

    if (msgctl(msgid, IPC_RMID, NULL) == -1) {
        perror("msgctl");
        exit(EXIT_FAILURE);
    }

    return 0;
}
```

Output:



Code:

```
GNU nano 7.2                                                    qu
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>

struct message {
    long msg_type;
    char msg_text[100];
};

int main() {
    key_t key;
    int msgid;
    struct message msg;

    key = ftok("/tmp", 'a');
    msgid = msgget(key, 0666 | IPC_CREAT);

    strcpy(msg.msg_text, "Hello from the sender!");
    msg.msg_type = 1;

    if (msgsnd(msgid, &msg, sizeof(msg.msg_text), 0) == -1) {
        perror("msgsnd");
        exit(EXIT_FAILURE);
    }

    printf("Message sent: %s\n", msg.msg_text);

    if (msgrcv(msgid, &msg, sizeof(msg.msg_text), 1, 0) == -1) {
        perror("msgrcv");
        exit(EXIT_FAILURE);
    }
```

```
        exit(EXIT_FAILURE);
    }

    printf("Message received: %s\n", msg.msg_text);

    if (msgctl(msgid, IPC_RMID, NULL) == -1) {
        perror("msgctl");
        exit(EXIT_FAILURE);
    }

    return 0;
}
```

Submitted By	Submitted to
MEDHANSH ALOK	ASHISH SIR
VARSHA	