**Operating Systems – Lab 06**

**Name:** Hafsa Salman

**Roll no.** 22K-5161

Experiment 6A

**Task no. 01**

Compile and execute the above code. Open a separate terminal window to note the process ids using in this code.

Code:

//Hafsa Salman

//22K-5161

//Experiment 6A: Task no. 01

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<sys/types.h>

#define BUFFER\_SIZE 25

#define READ\_END 0

#define WRITE\_END 1

int main(void)

{

char write\_msg[BUFFER\_SIZE] = "Greetings";

char read\_msg[BUFFER\_SIZE];

int fd[2];

pid\_t pid;

if (pipe(fd) == -1)

{

fprintf(stderr, "Pipe Failed!\n");

return 1;

}

pid = fork();

if (pid < 0)

{

fprintf(stderr, "Fork Failed!\n");

return 1;

}

if (pid > 0)

{

close(fd[READ\_END]);

write(fd[WRITE\_END], write\_msg, strlen(write\_msg)+1);

close(fd[WRITE\_END]);

}

else

{

close(fd[WRITE\_END]);

read(fd[READ\_END], read\_msg, BUFFER\_SIZE);

printf("Read %s\n", read\_msg);

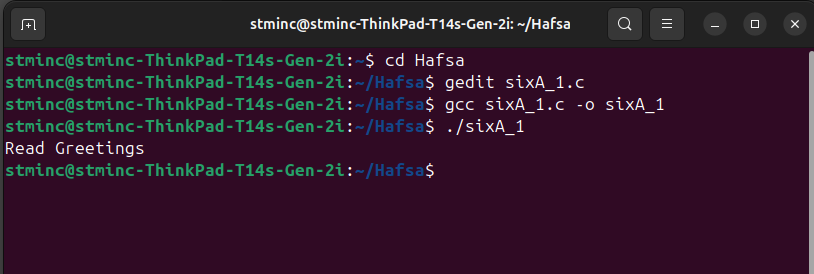
close(fd[READ\_END]);

}

return 0;

}

Output:



**Task no. 02**

Modify the code such that it tasks write\_msg value from user terminal.

Code:

//Hafsa Salman

//22K-5161

//Experiment 6A: Task no. 02

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<sys/types.h>

#define BUFFER\_SIZE 25

#define READ\_END 0

#define WRITE\_END 1

int main(void)

{

char write\_msg[BUFFER\_SIZE];

char read\_msg[BUFFER\_SIZE];

int fd[2];

printf("Enter message: ");

fgets(write\_msg, BUFFER\_SIZE, stdin);

pid\_t pid;

if (pipe(fd) == -1)

{

fprintf(stderr, "Pipe Failed!\n");

return 1;

}

pid = fork();

if (pid < 0)

{

fprintf(stderr, "Fork Failed!\n");

return 1;

}

if (pid > 0)

{

close(fd[READ\_END]);

write(fd[WRITE\_END], write\_msg, strlen(write\_msg)+1);

close(fd[WRITE\_END]);

}

else

{

close(fd[WRITE\_END]);

read(fd[READ\_END], read\_msg, BUFFER\_SIZE);

printf("Read %s\n", read\_msg);

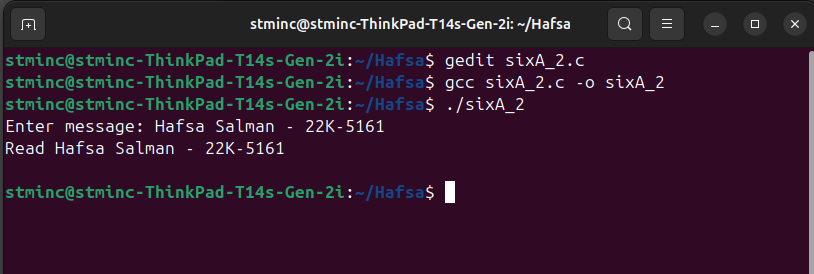
close(fd[READ\_END]);

}

return 0;

}

Output:



Experiment 6B

**Task no. 01**

Compile and execute the producer code above that uses named pipe. Correct errors, if any.

Code:

//Hafsa Salman

//22K-5161

//Experminent 6B: Task no. 01

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<string.h>

#define FIFO\_FILE "/tmp/myfifo"

int main()

{

int fd;

char buffer[BUFSIZ];

ssize\_t num\_bytes;

mkfifo(FIFO\_FILE, 0666);

fd = open(FIFO\_FILE, O\_WRONLY);

if (fd == -1)

{

perror("open");

exit(EXIT\_FAILURE);

}

while (1)

{

printf("Producer: Enter a message (or 'exit' to quit): ");

fgets(buffer, BUFSIZ, stdin);

num\_bytes = write(fd, buffer, strlen(buffer));

if (num\_bytes == -1)

{

perror("write");

exit(EXIT\_FAILURE);

}

if (strncmp(buffer, "exit", 4) == 0)

{

break;

}

}

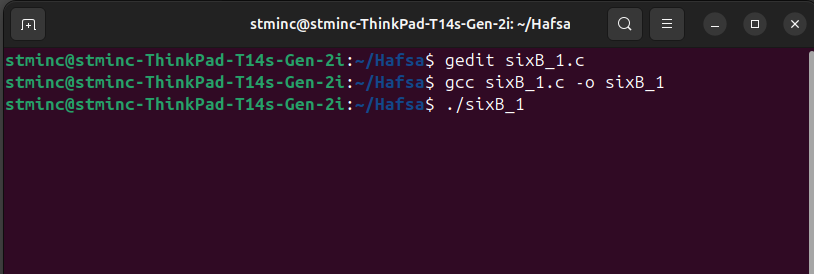
close(fd);

unlink(FIFO\_FILE);

return 0;

}

Output:

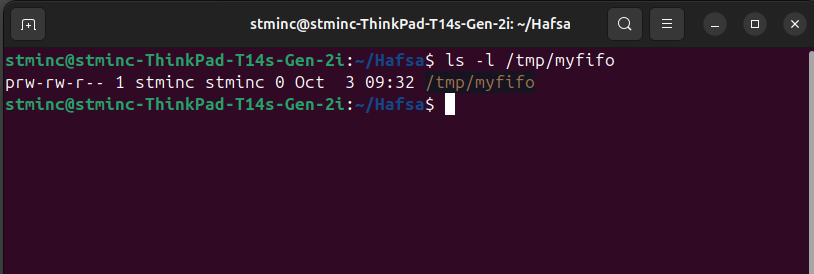


//Other screenshots under Task no. 03 (Consumer code)

**Task no. 02**

Carefully study the producer code and verify that the FIFO\_FILE has been created.

Output:



**Task no. 03**

Now, write consumer code for the above producer using the following hints:

• Open FIFO\_FILE in read only mode using O\_RDONLY

• Consumer read the data using num\_bytes = read(fd, buffer, BUFSIZ);

• Display the data sent by the producer using

printf("Consumer: Received message: %s", buffer);

Code:

//Hafsa Salman

//22K-5161

//Experiment 6B: Task no. 03

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include<string.h>

#define FIFO\_FILE "/tmp/myfifo"

int main()

{

int fd;

char buffer[BUFSIZ];

ssize\_t num\_bytes;

fd = open(FIFO\_FILE, O\_RDONLY);

if (fd == -1)

{

perror("open");

exit(EXIT\_FAILURE);

}

while (1)

{

num\_bytes = read(fd, buffer, BUFSIZ);

if (num\_bytes == -1)

{

perror("read");

break;

}

else if (num\_bytes == 0)

{

printf("Producer exited the FIFO. Exiting...\n");

break;

}

else

{

buffer[num\_bytes] = '\0';

printf("Consumer - Message recieved: %s", buffer);

if (strncmp(buffer, "exit", 4) == 0)

{

printf("Exit");

break;

}

}

}

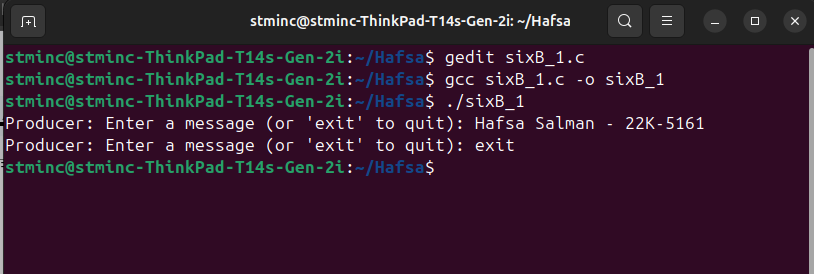
close(fd);

return 0;

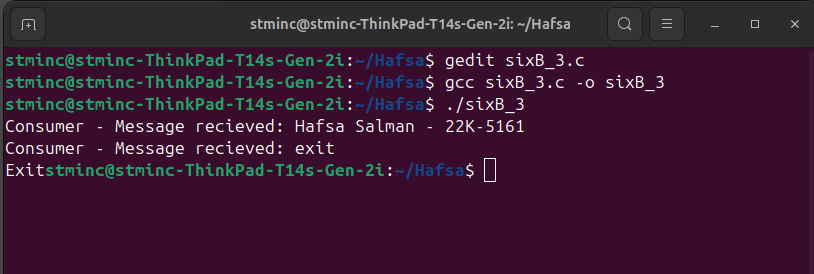
}

Output:

Producer terminal:



Consumer Terminal:



Experiment 6C

**Task no. 01**

Compile and execute producer code.

Code:

//Hafsa Salman

//22K-5161

//Experiment 6C: Task no. 01

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<fcntl.h>

#include<sys/shm.h>

#include<sys/stat.h>

#include<sys/mman.h>

#include<unistd.h>

int main()

{

const int SIZE = 4096;

const char \*name = "OS";

const char \*message\_0 = "Hello";

const char \*message\_1 = "World!";

int fd;

char \*ptr;

fd = shm\_open(name, O\_CREAT | O\_RDWR, 0666);

ftruncate(fd, SIZE);

ptr = (char \*)mmap(0, SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

sprintf(ptr, "%s", message\_0);

ptr += strlen(message\_0);

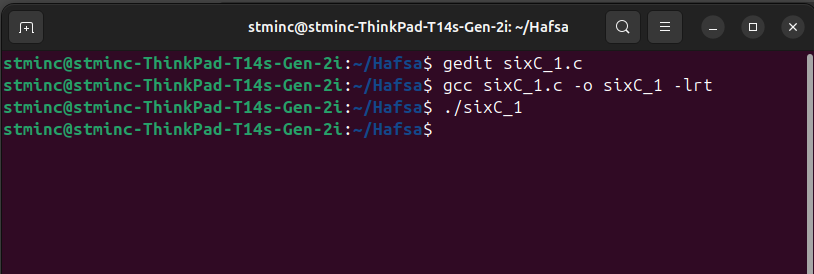
sprintf(ptr, "%s", message\_1);

ptr += strlen(message\_1);

return 0;

}

Output:



**Task no. 02**

Compile and execute the consumer code. It will give a run-time error. Why?

Code:

//Hafsa Salman

//22K-5161

//Experiment 6C: Task no. 02

#include<stdio.h>

#include<stdlib.h>

#include<fcntl.h>

#include<sys/shm.h>

#include<sys/stat.h>

#include<sys/mman.h>

int main()

{

const int SIZE = 4096;

const char \*name = "OS";

int fd;

char \*ptr;

fd = shm\_open(name, O\_RDONLY, 0666);

ptr = (char \*)mmap(0, SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

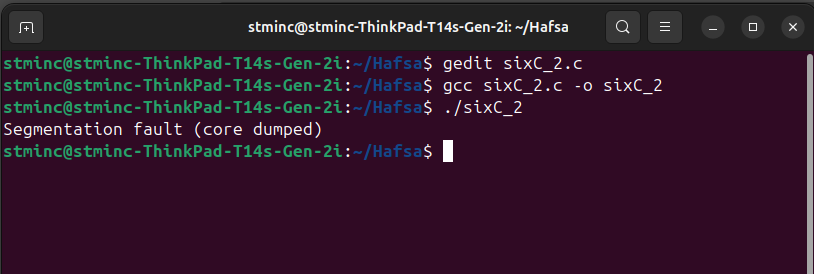
printf("%s", (char \*)ptr);

shm\_unlink(name);

return 0;

}

Output:



Reason:

It is mapping file writable without having write permission on the file descriptor.

**Task no. 03**

Now modify the producer code such that the producer and the consumer code run concurrently as two processes. Now the consumer code will print the message.

Code:

//Hafsa Salman

//22K-5161

//Experiment 6C: Task no. 03

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<fcntl.h>

#include<sys/shm.h>

#include<sys/stat.h>

#include<sys/mman.h>

#include<unistd.h>

int main()

{

const int SIZE = 4096;

const char \*name = "OS";

const char \*message\_0 = "Hello";

const char \*message\_1 = "World!";

int fd;

char \*ptr;

fd = shm\_open(name, O\_CREAT | O\_RDWR, 0666);

if (fd == -1)

{

perror("shm\_error");

exit(1);

}

ftruncate(fd, SIZE);

ptr = (char \*)mmap(0, SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

if (ptr == MAP\_FAILED)

{

perror("mmap");

exit(1);

}

sprintf(ptr, "%s", message\_0);

ptr += strlen(message\_0);

sprintf(ptr, "%s", message\_1);

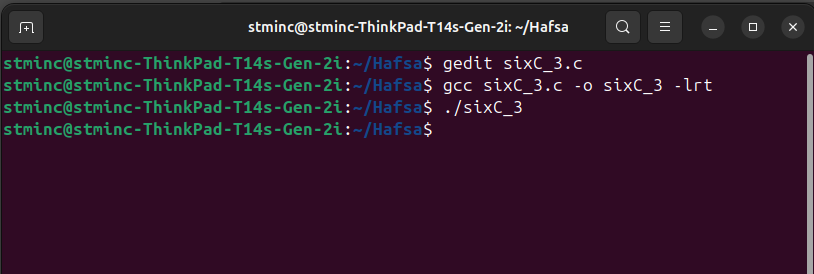
ptr += strlen(message\_1);

return 0;

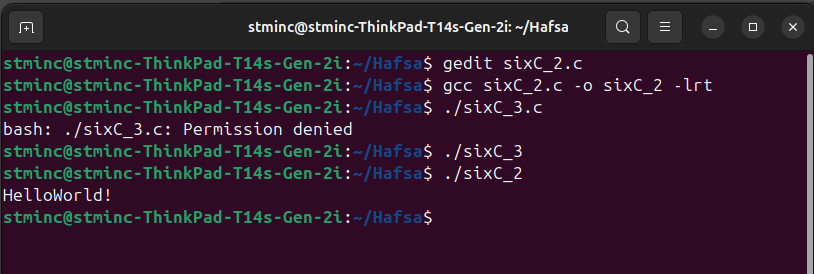
}

Output:

Producer Terminal:



Consumer Terminal:



**Task no. 04:**

Now modify the consumer code to print the second message.

Code:

//Hafsa Salman

//22K-5161

//Experiment 6C: Task no. 04

#include<stdio.h>

#include<stdlib.h>

#include<fcntl.h>

#include<sys/shm.h>

#include<sys/stat.h>

#include<sys/mman.h>

#include<unistd.h>

int main()

{

const int SIZE = 4096;

const char \*name = "OS";

int fd;

char \*ptr;

fd = shm\_open(name, O\_RDONLY, 0666);

ptr = (char \*)mmap(0, SIZE, PROT\_READ, MAP\_SHARED, fd, 0);

printf("%s \n", (char \*)ptr);

shm\_unlink(name);

return 0;

}

Output:

