#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define MSG\_LEN 64

int main()

{

int result;

int fd[2];

char message[MSG\_LEN];

char recvd\_msg[MSG\_LEN];

result = pipe (fd);

//Creating a pipe

//fd[0] is for reading and fd[1] is for writing

if (result < 0)

{

perror("pipe ");

exit(1);

}

strncpy(message,"Linux World!! ",MSG\_LEN);

result=write(fd[1],message,strlen(message));

if (result< 0)

{

perror("write");

exit(2);

}

strncpy(message,"Understanding ",MSG\_LEN);

result=write(fd[1],message,strlen(message));

if (result < 0)

{

perror("write");

exit(2);

}

strncpy(message,"Concepts of ",MSG\_LEN);

result=write(fd[1],message,strlen(message));

if (result < 0)

{

perror("write");

exit(2);

}

strncpy(message,"Piping ", MSG\_LEN);

result=write(fd[1],message,strlen(message));

if (result < 0)

{

perror("write");

exit(2);

}

result=read(fd[0],recvd\_msg,MSG\_LEN);

if (result < 0)

{

perror("read");

exit(3);

}

printf("%s\n",recvd\_msg);

return 0;

}

**OUTPUT :**

**Linux World!! Understanding concepts of piping.**

FIFO :

// C program to implement one side of FIFO

// This side writes first, then reads

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

int fd;

// FIFO file path

char \* myfifo = "/tmp/myfifo";

// Creating the named file(FIFO)

// mkfifo(<pathname>, <permission>)

mkfifo(myfifo, 0666);

char arr1[80], arr2[80];

while (1)

{

// Open FIFO for write only

fd = open(myfifo, O\_WRONLY);

// Take an input arr2ing from user.

// 80 is maximum length

fgets(arr2, 80, stdin);

// Write the input arr2ing on FIFO

// and close it

write(fd, arr2, strlen(arr2)+1);

close(fd);

// Open FIFO for Read only

fd = open(myfifo, O\_RDONLY);

// Read from FIFO

read(fd, arr1, sizeof(arr1));

// Print the read message

printf("User2: %s\n", arr1);

close(fd);

}

return 0;

}

// C program to implement one side of FIFO

// This side reads first, then reads

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

int fd1;

// FIFO file path

char \* myfifo = "/tmp/myfifo";

// Creating the named file(FIFO)

// mkfifo(<pathname>,<permission>)

mkfifo(myfifo, 0666);

char str1[80], str2[80];

while (1)

{

// First open in read only and read

fd1 = open(myfifo,O\_RDONLY);

read(fd1, str1, 80);

// Print the read string and close

printf("User1: %s\n", str1);

close(fd1);

// Now open in write mode and write

// string taken from user.

fd1 = open(myfifo,O\_WRONLY);

fgets(str2, 80, stdin);

write(fd1, str2, strlen(str2)+1);

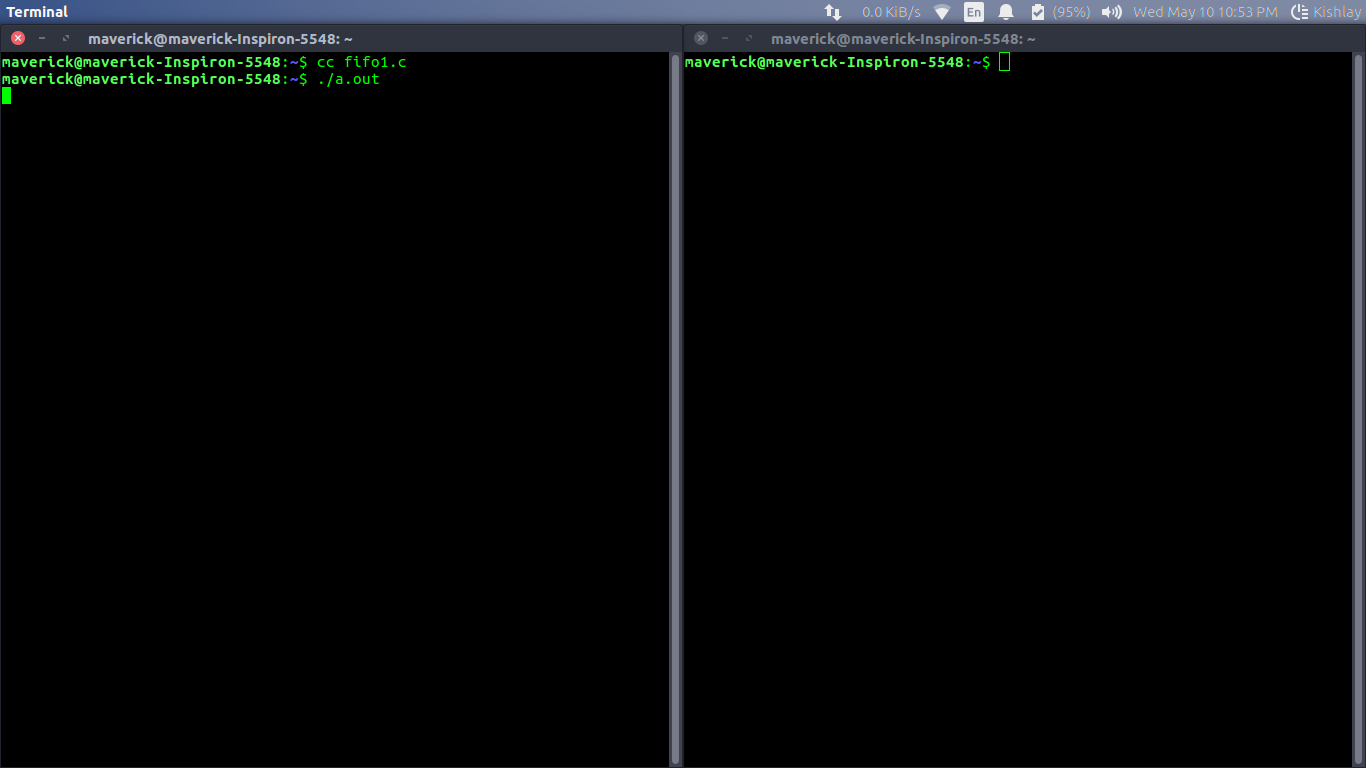
close(fd1);

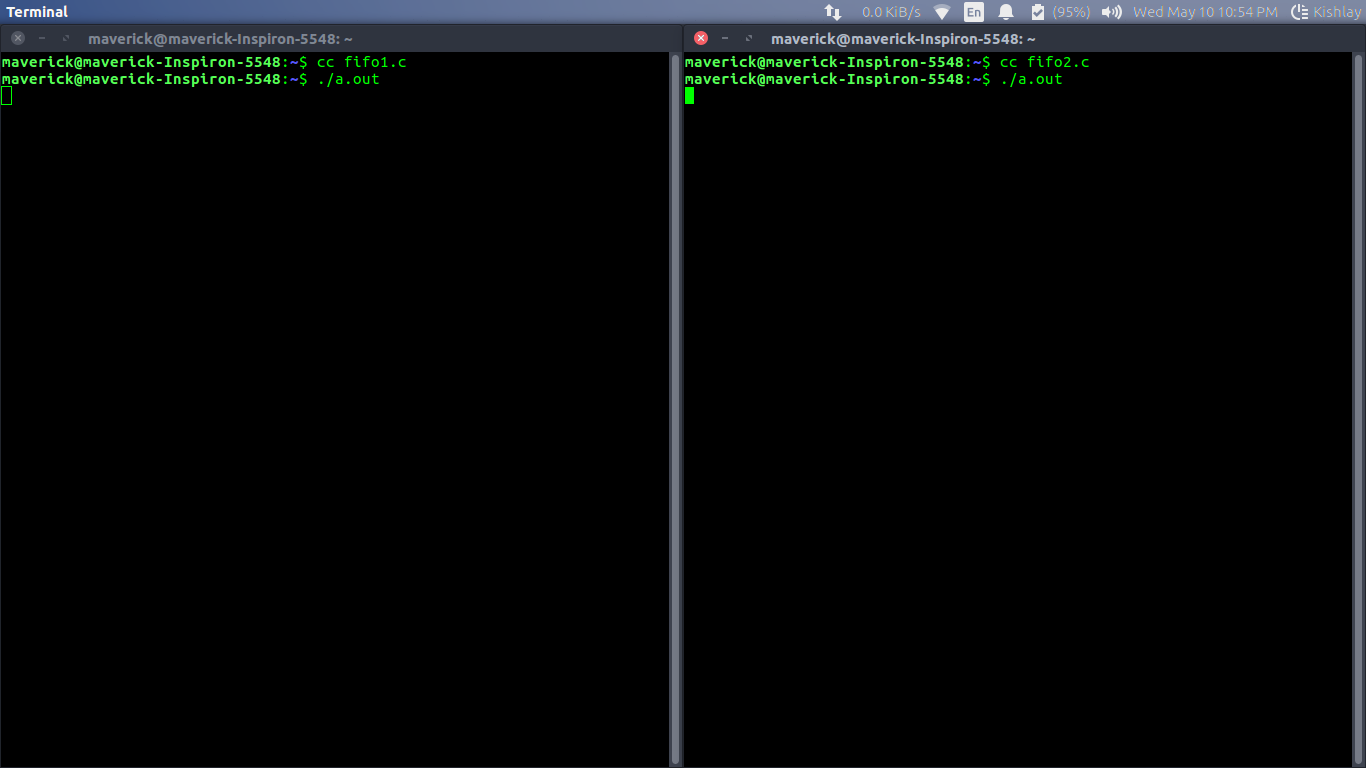
}

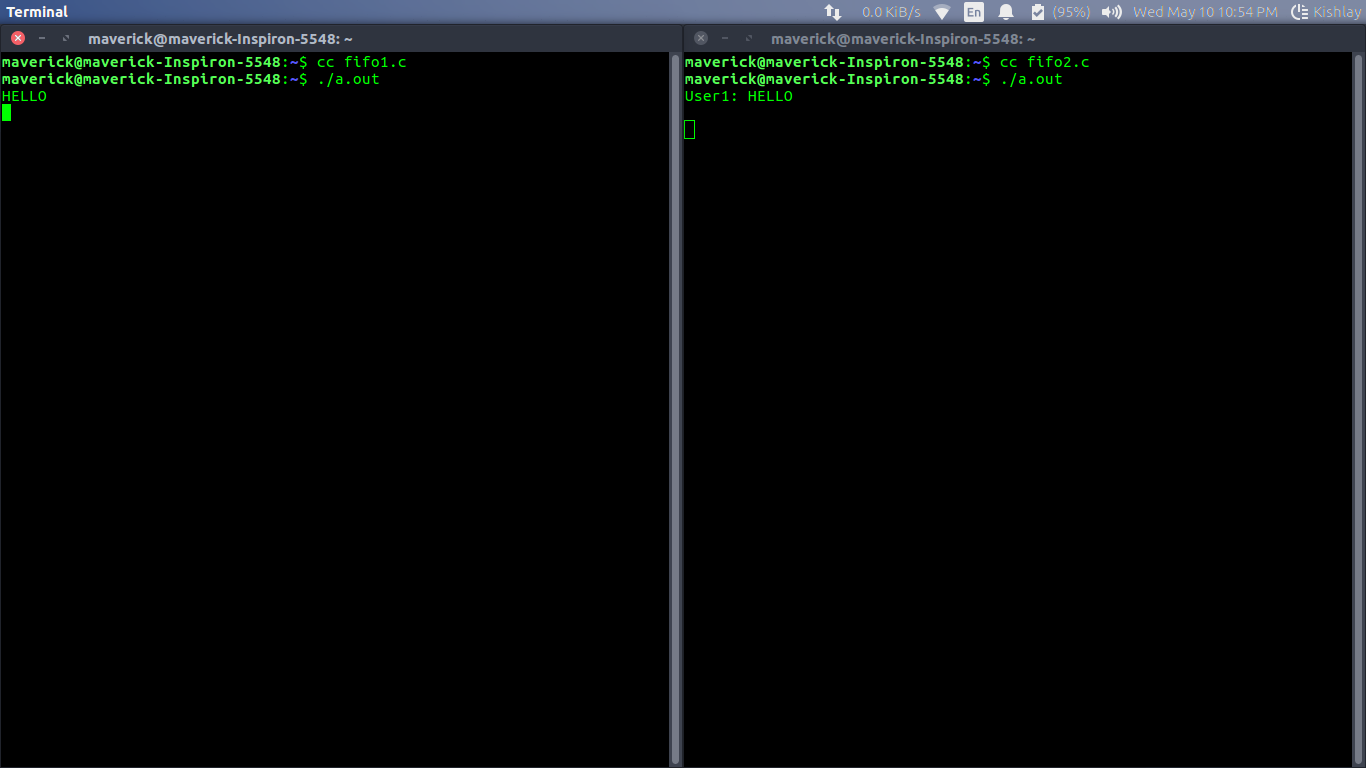
return 0;

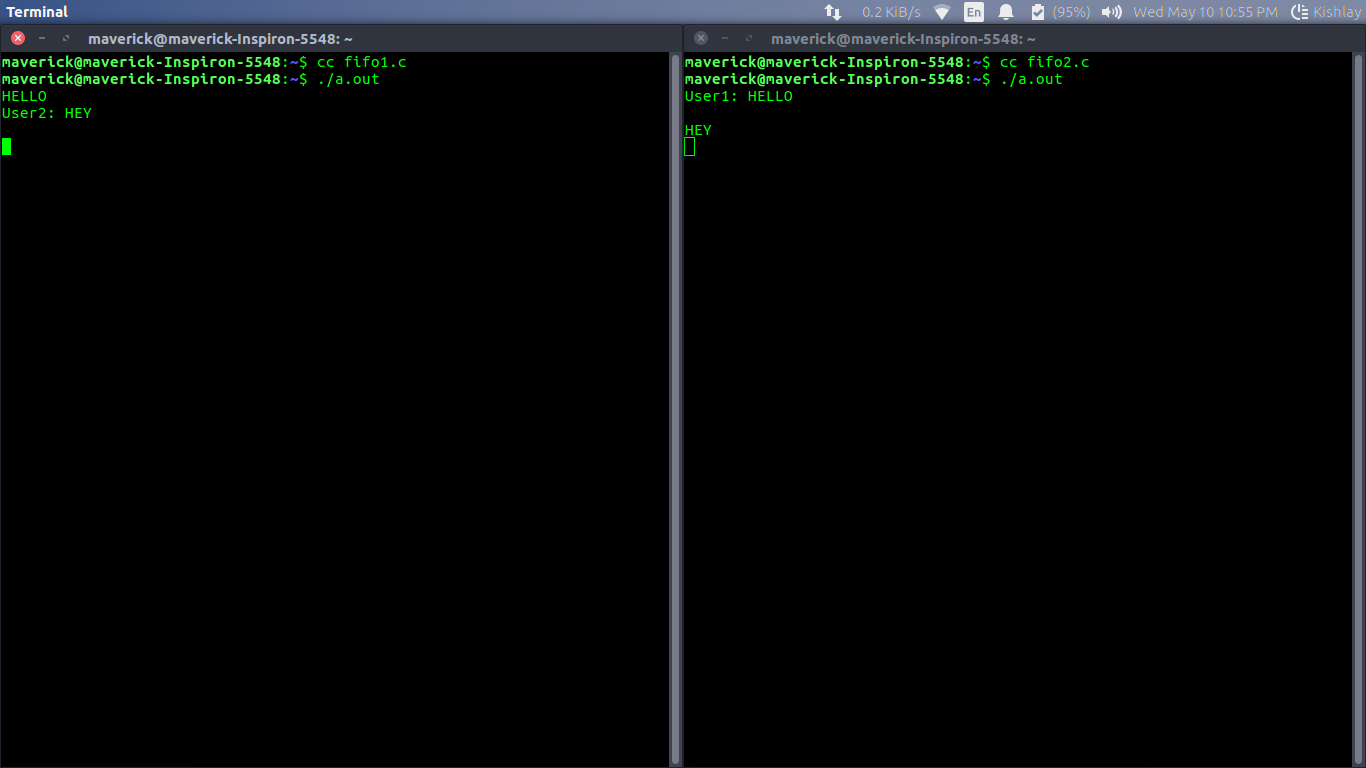
}

Output :







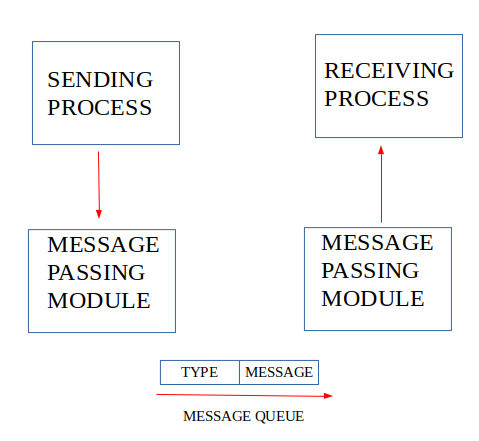


**Message Queue:**

A message queue is a linked list of messages stored within the kernel and identified by a message queue identifier. A new queue is created or an existing queue opened by **msgget()**.

New messages are added to the end of a queue by **msgsnd()**.

 when the message is added to a queue. Messages are fetched from a queue by **msgrcv()**



* **ftok()**: is use to generate a unique key.
* **msgget()**: either returns the message queue identifier for a newly created message queue or returns the identifiers for a queue which exists with the same key value.
* **msgsnd()**: Data is placed on to a message queue by calling msgsnd().
* **msgrcv()**: messages are retrieved from a queue.
* **msgctl()**: It performs various operations on a queue. Generally it is use to destroy message queue.

**// C Program for Message Queue (Writer Process)**

**#include <stdio.h>**

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

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

**#define MAX 10**

**// structure for message queue**

**struct mesg\_buffer {**

**long mesg\_type;**

**char mesg\_text[100];**

**} message;**

**int main()**

**{**

**key\_t key;**

**int msgid;**

**// ftok to generate unique key**

**key = ftok("progfile", 65);**

**// msgget creates a message queue**

**// and returns identifier**

**msgid = msgget(key, 0666 | IPC\_CREAT);**

**message.mesg\_type = 1;**

**printf("Write Data : ");**

**fgets(message.mesg\_text,MAX,stdin);**

**// msgsnd to send message**

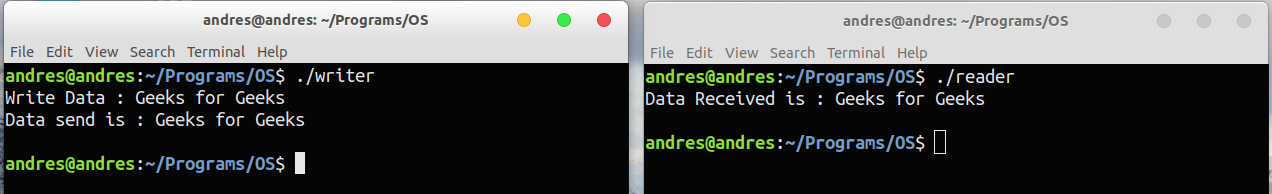
**msgsnd(msgid, &message, sizeof(message), 0);**

**// display the message**

**printf("Data send is : %s \n", message.mesg\_text);**

**return 0;**

**}**



**// C Program for Message Queue (Reader Process)**

**#include <stdio.h>**

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

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

**// structure for message queue**

**struct mesg\_buffer {**

**long mesg\_type;**

**char mesg\_text[100];**

**} message;**

**int main()**

**{**

**key\_t key;**

**int msgid;**

**// ftok to generate unique key**

**key = ftok("progfile", 65);**

**// msgget creates a message queue**

**// and returns identifier**

**msgid = msgget(key, 0666 | IPC\_CREAT);**

**// msgrcv to receive message**

**msgrcv(msgid, &message, sizeof(message), 1, 0);**

**// display the message**

**printf("Data Received is : %s \n", message.mesg\_text);**

**// to destroy the message queue**

**msgctl(msgid, IPC\_RMID, NULL);**

**return 0;**

**}**