آزمایش 4

فاطمه راق 40013045

حامد فلاحی 9931043

خلاصه

بطور کلی، فرآیند هایی که همزمان در سیستم عامل اجرا می شوند، میتوانند مستقل یا همکار باشند. یک فرآیند مستقل تحت تاثیر فرآیند هاي در حال اجرا در سیستم نیست و بر آنها تاثیري نمی گذارد. فرآیند مستقل با هیچ فرآیند دیگري داده به اشتراک نمی گذارد. در مقابل فرآیند همکار با فرآیند هاي دیگر در حال اجرا در سیستم داده به اشتراک می گذارد و میتواند از آنها تاثیر بگیرد و یا تاثیر بگذارد.

حال ما برای ارتباط فرایند ها با یکدیگر، 3 حالت داریم که بر اساس آنها، ارتباطات بین فرایند هارا پیش می بریم:

Shared memory

Message passing

Pipeline

حال در 3 بخش آورده شده در دستورکار، باتوجه به نیازهرکدام که گفته شده، از آن استفاده میکنیم.

(HW1

در بخش 1، از ما خواسته شده که محیطی آماده کنیم که دو فرآیند در آن وجود داشته باشند و از روش حافظه مشترک براي ارتباط استفاده کنند. حال برای اینکه از روش حافظه مشترک برای دو فرایند استفاده کنیم ابتدا نیازمندیم که 2 برنامه c ایجاد کنیم که یکی بنویسد و دیگری بخواند. همچنین برای آنهای باید از کتابخانه های حالت حافظه مشترک استفاده کرد و همانطور که در عکس ها میبینیم در کد آمده است.سپس در کد اول با استفاده از متد ()ftok یک کلید ایجاد کنیم و از آن برای ارتباطات استفاده کنیم. سپس کلید تولید شده را طبق گفته دستور کار به ()shmget میدهیم. همینطور از ()shmat نیز استفاده میکنیم . سپس در انتهای کد، رشته را خوانده و در متغیر کپی میکنیم. در کد دوم نیز همانند کد اول استو مقدار نهایی را خوانده و برای ما چاپ میکند.
عکس نتایج و کد نیز در پایین فرستاده شده است:

2.c - Notepad

File Edit View

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
      key_t key = ftok("File",65);
      void *sharedM;
      char str[1000];
      int sharedM_id;
      sharedM_id=shmget(key, 1024, 0444);
      sharedM=shmat(sharedM_id,NULL,0);
      printf("Address: %p\n",sharedM);
      printf("The value stored in this address is: %s\n",(char *)sharedM);
      return 0;
}
```

Ln 1, Col 1 100% Unix (LF) UTF-8

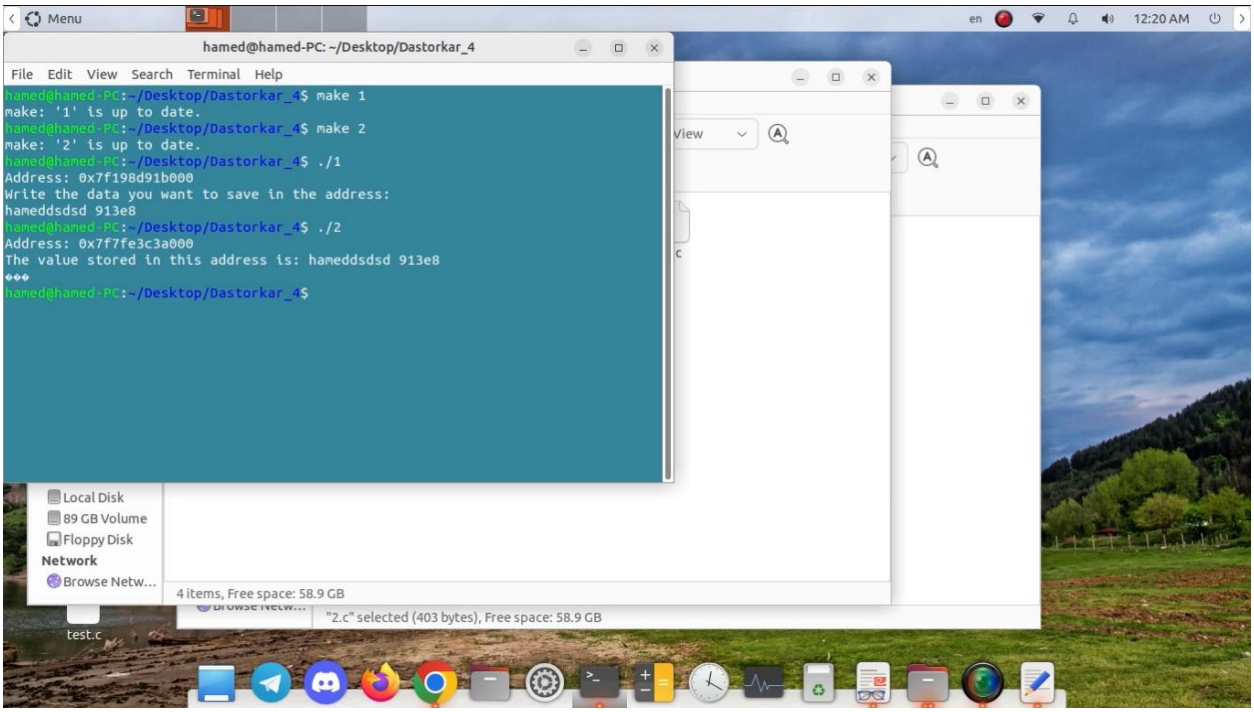
1.c - Notepad

File Edit View

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
    key_t key = ftok("File",65);
      char str[1000];
      void *sharedM;
      int sharedM_id;
      sharedM_id=shmget(key, 1024, 0666|IPC_CREAT);
      sharedM=shmat(sharedM_id,NULL,0);
      printf("Address: %p\n",sharedM);
      printf("Write the data you want to save in the address:\n");
      read(0,str,1000);
      strcpy(sharedM,str);
      return 0;
}
```

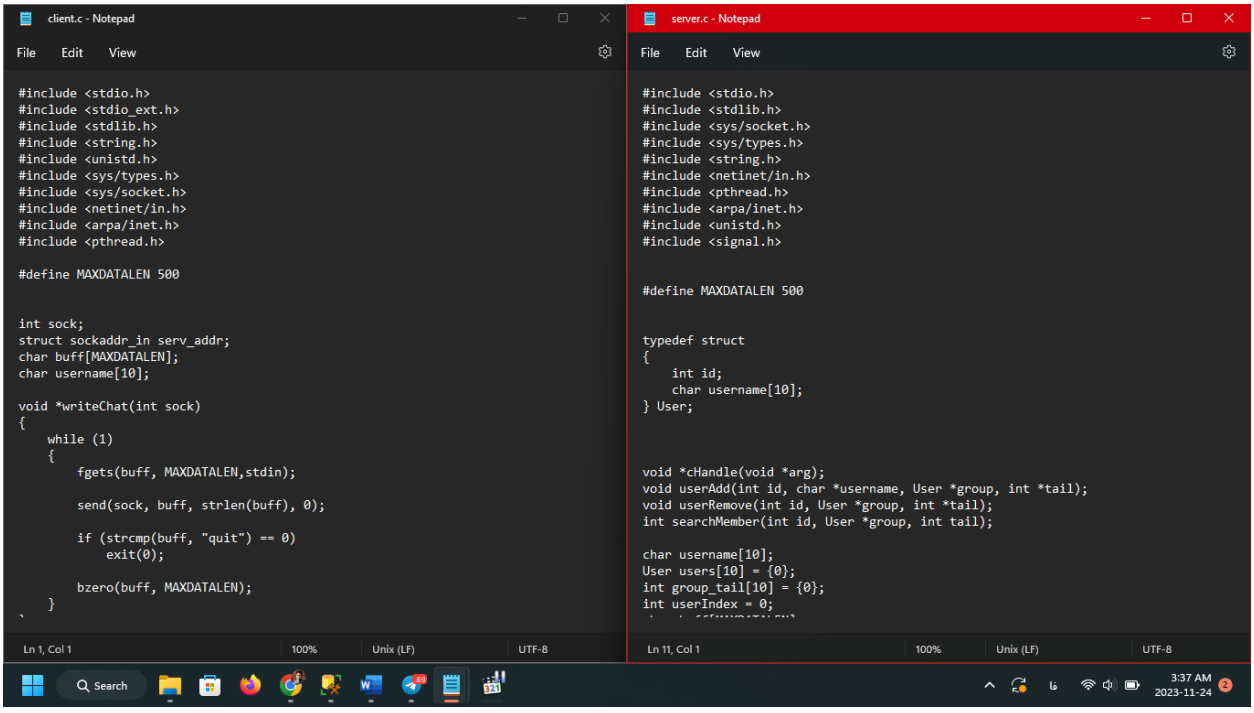
Ln 1, Col 1 100% Unix (LF) UTF-8

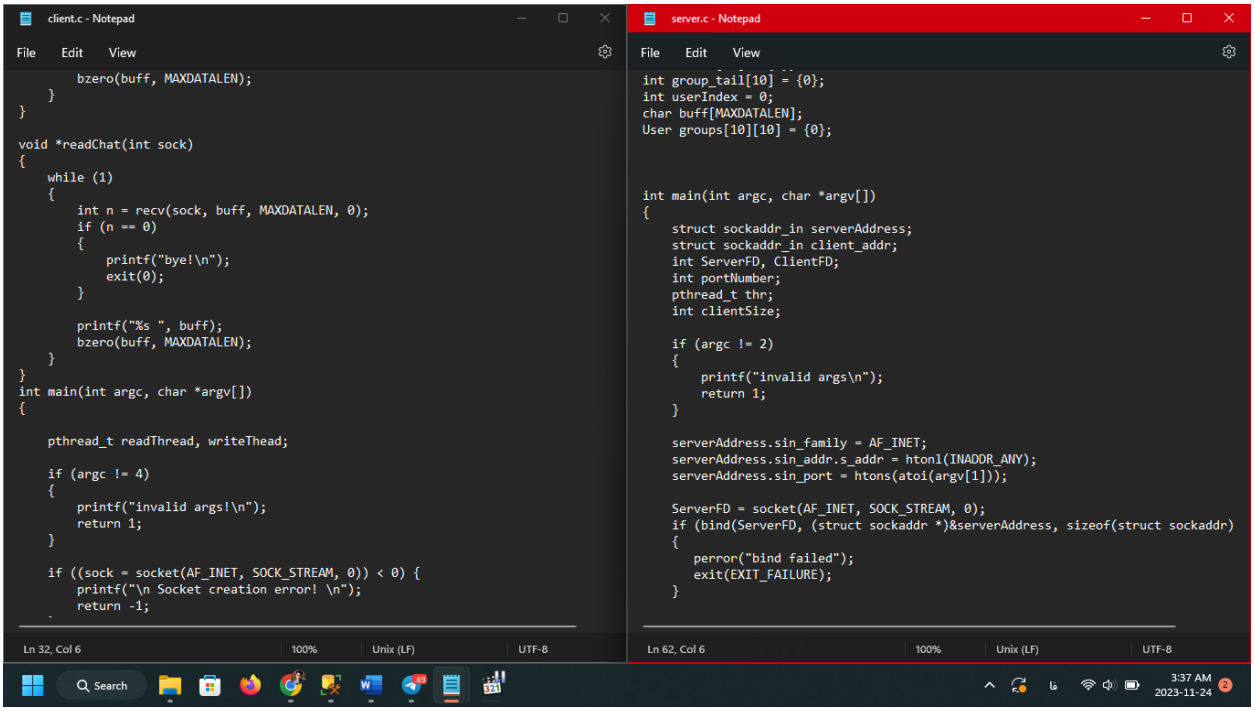
Search 2:53 AM 2023-11-24

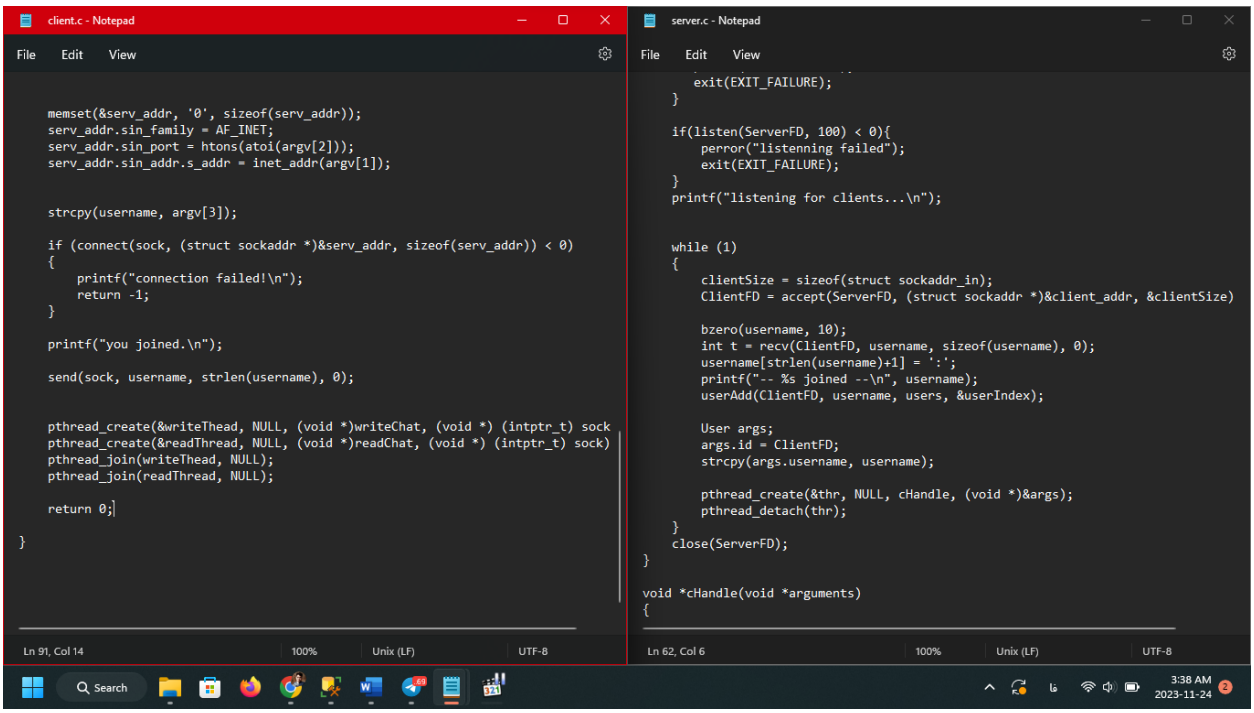


(HW2

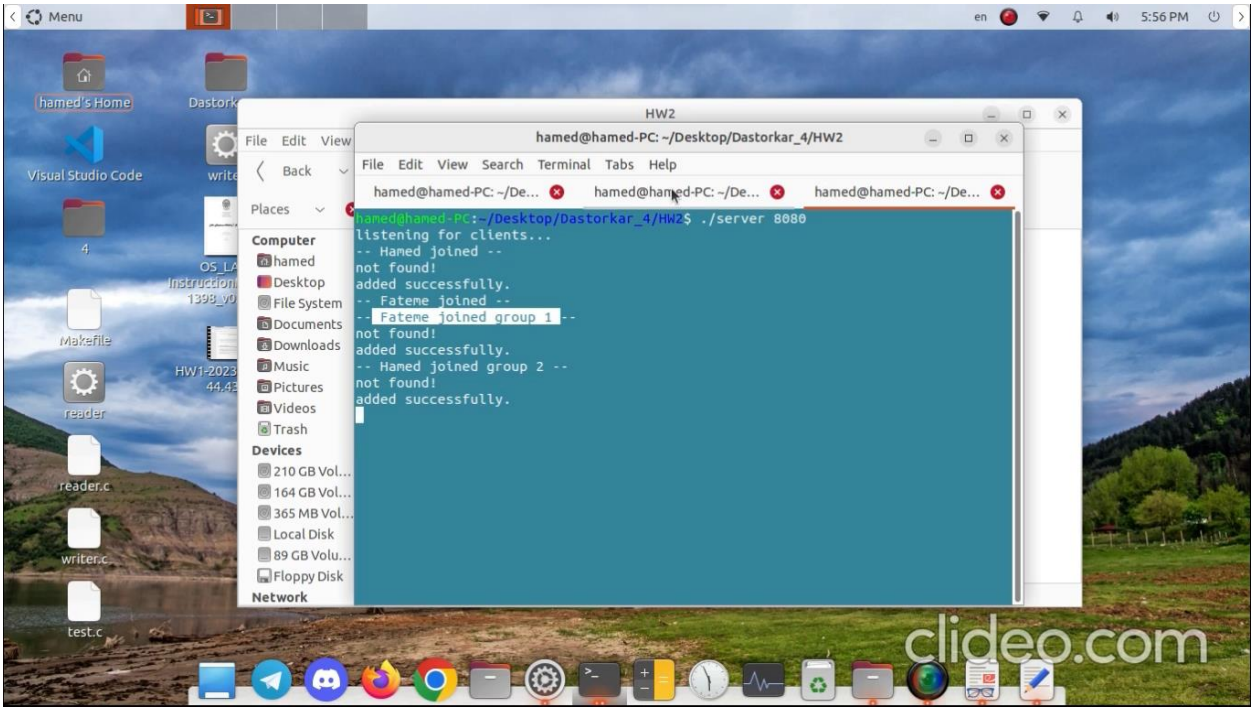
بخش 2 از ما خواستار است که یک برنامه کاربردي گفتگو را به وسیله ي زبان C پیاده سازي کنیم. این برنامه دو بخش دارد: سرور و کاربر. سرور اطلاعات کاربران را نگه داري می کند و وقتی پیامی توسط یک کاربر فرستاده می شود با توجه به گروه مورد نظر این پیام را بین کاربران پخش میکند. راه ارتباط کابران با یکدیگر، وارد شدن به یک گروه چت است. برای ایجاد این برنامه ما نیاز به ساخت دو برنامه c هستیم که یکی سرور و دیگری کلاینت باشد. سپس کد سرور را باید باتوجه به متد ها و ویژگی های و نحوه اجرا و روند کد که در دستور کار گفته شده است، بسازیم و آن را آپدیت کنیم. به همین دلیل پس از استفاده از نمونه کد سرور داده شده در دستور کار ما طبق عکس داده شده نیاز داریم به متغیرهای مختلف برای ساختن اکانت های مختلف با آی دی های متفاوت و همچنین گروه های مختلف که هر گروه شامل چندین اکانت شود. پس از این کار باید متد اصلی برنامه را که روند اجرای برنامه را تعیین میکند، طبق متغیر ها آپدیت کرده و در اخر یک ریسمان برای انجام کار اضافه کنیم. همچنین کد کلاینت تقریبا با سرور یکی است و متد های تعریف شده سرور را ندارد. همچنین در آخر چون که کلاینت هم میخواند و هم مینویسد، باید دو ریسمان برای آن درنظر بگیریم تا مشکل برطرف شود. سپس هردو کد را طبق عکس های ارسالی در پایین صفحه کامپایل و اجرا میکنیم:

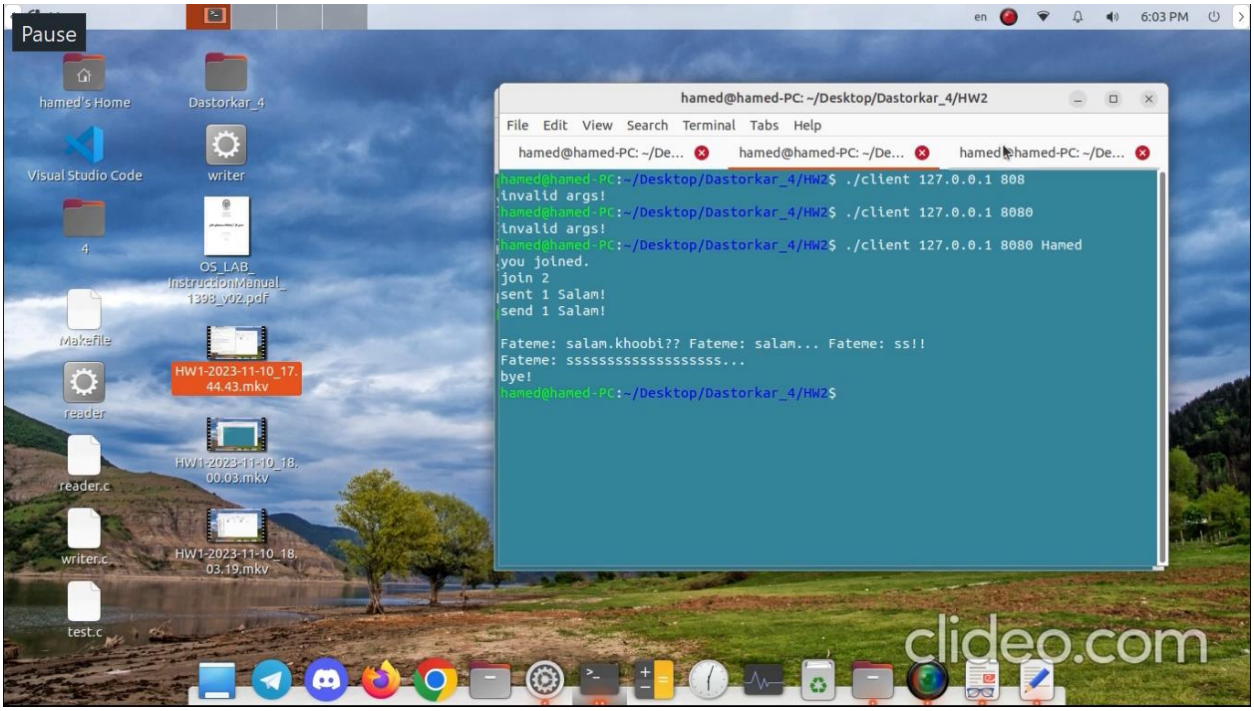


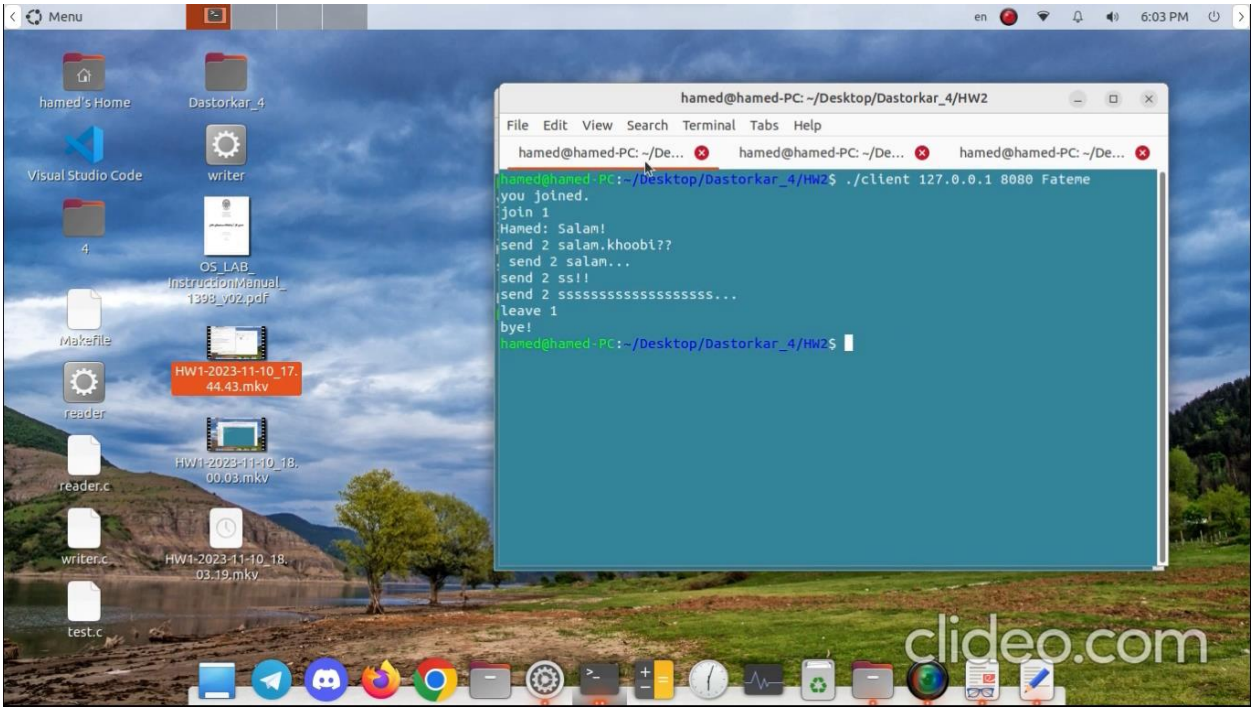


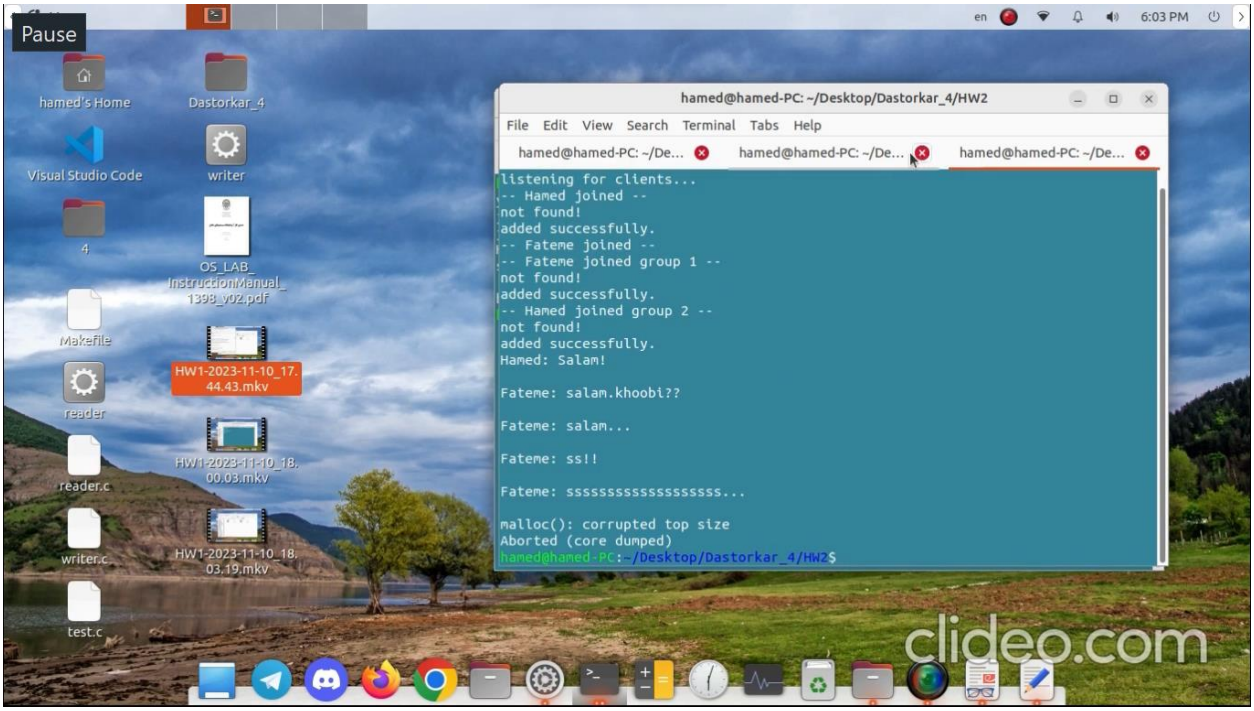


(لطفا ادامه کد سرور را درصورت خواستن از بخش کد ها مشاهده کنید)


Menu
en
5:56 PM
hamed's Home
Visual Studio Code
4
Makefile
reader
reader.c
writer.c
test.c
HW2
File Edit View
Back
Places
Computer
hamed
Desktop
File System
Documents
Downloads
Music
Pictures
Videos
Trash
Devices
210 GB Vol...
164 GB Vol...
365 MB Vol...
Local Disk
89 GB Volu...
Floppy Disk
Network
hamed@hamed-PC: ~/Desktop/Dastorkar_4/HW2
File Edit View Search Terminal Tabs Help
hamed@hamed-PC: ~/De...
hamed@hamed-PC: ~/De...
hamed@hamed-PC: ~/De...
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW2$ ./server 8080
listening for clients...
-- Hamed joined --
not found!
added successfully.
-- Fateme joined --
-- Fateme joined group 1 --
not found!
added successfully.
-- Hamed joined group 2 --
not found!
added successfully.
clideo.com


Pause
en
6:03 PM
hamed's Home
Visual Studio Code
4
Makefile
reader
reader.c
writer.c
test.c
Dastorkar_4
writer
OS_LAB_InstructionManual_1398_v02.pdf
HW1-2023-11-10_17.44.43.mkv
HW1-2023-11-10_18.00.03.mkv
HW1-2023-11-10_18.03.19.mkv
hamed@hamed-PC: ~/Desktop/Dastorkar_4/HW2
File Edit View Search Terminal Tabs Help
hamed@hamed-PC: ~/De...
hamed@hamed-PC: ~/De...
hamed@hamed-PC: ~/De...
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW2$ ./client 127.0.0.1 808
invalid args!
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW2$ ./client 127.0.0.1 8080
invalid args!
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW2$ ./client 127.0.0.1 8080 Hamed
you joined.
join 2
sent 1 Salam!
send 1 Salam!
Fateme: salam.khoobi?? Fateme: salam... Fateme: ss!!
Fateme: sssssssssssssssss...
bye!
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW2$
clideo.com

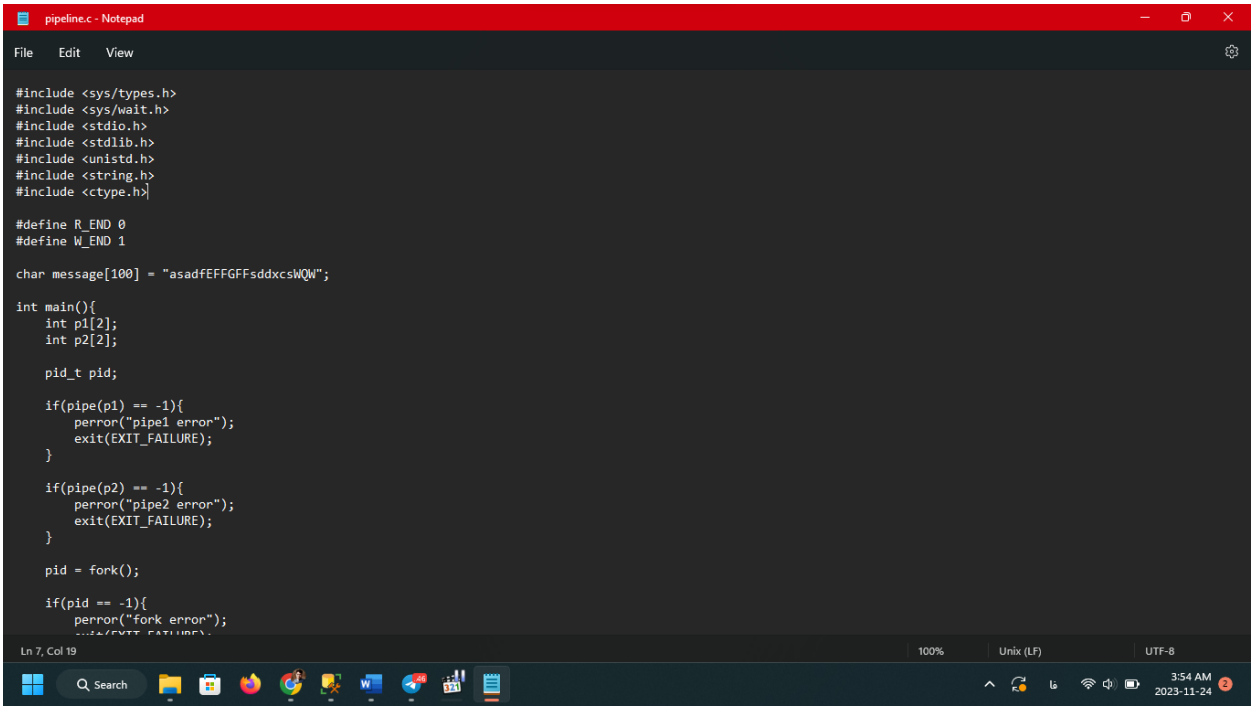

Menu
en
6:03 PM
hamed's Home
Dastorkar_4
Visual Studio Code
writer
4
OS_LAB_InstructionManual_1398_v02.pdf
Makefile
HW1-2023-11-10_17.44.43.mkv
reader
HW1-2023-11-10_18.00.03.mkv
reader.c
writer.c
HW1-2023-11-10_18.03.19.mkv
test.c
hamed@hamed-PC: ~/Desktop/Dastorkar_4/HW2
File Edit View Search Terminal Tabs Help
hamed@hamed-PC: ~/De...
hamed@hamed-PC: ~/De...
hamed@hamed-PC: ~/De...
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW2$ ./client 127.0.0.1 8080 Fateme
you joined.
join 1
Hamed: Salam!
send 2 salam.khoobi??
send 2 salam...
send 2 ss!!
send 2 sssssssssssssssssss...
leave 1
bye!
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW2$
clideo.com

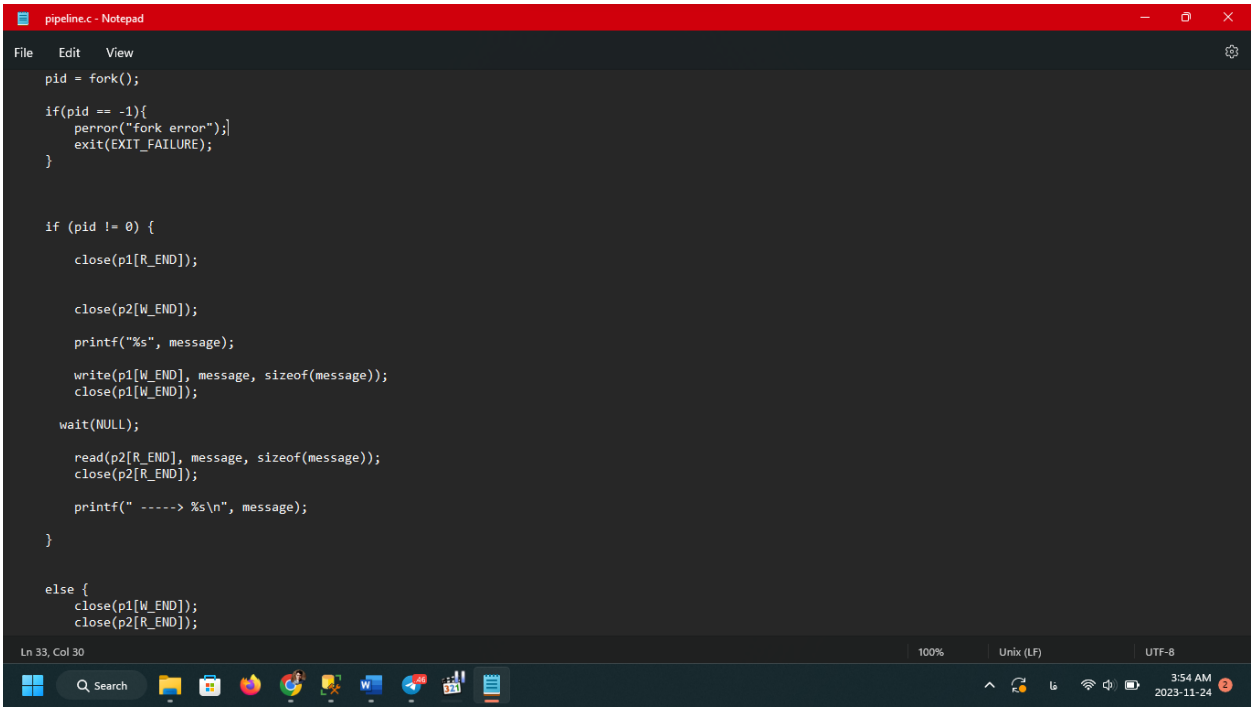

Pause
en
6:03 PM
hamed's Home
Dastorkar_4
Visual Studio Code
writer
4
OS_LAB_InstructionManual_1398_v02.pdf
Makefile
HW1-2023-11-10_17.44.43.mkv
reader
HW1-2023-11-10_18.00.03.mkv
reader.c
writer.c
HW1-2023-11-10_18.03.19.mkv
test.c
hamed@hamed-PC: ~/Desktop/Dastorkar_4/HW2
File Edit View Search Terminal Tabs Help
hamed@hamed-PC: ~/De...
hamed@hamed-PC: ~/De...
hamed@hamed-PC: ~/De...
listening for clients...
-- Hamed joined --
not found!
added successfully.
-- Fateme joined --
-- Fateme joined group 1 --
not found!
added successfully.
-- Hamed joined group 2 --
not found!
added successfully.
Hamed: Salam!
Fateme: salam.khoobi??
Fateme: salam...
Fateme: ss!!
Fateme: sssssssssssssssssss...
malloc(): corrupted top size
Aborted (core dumped)
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW2$
clideo.com

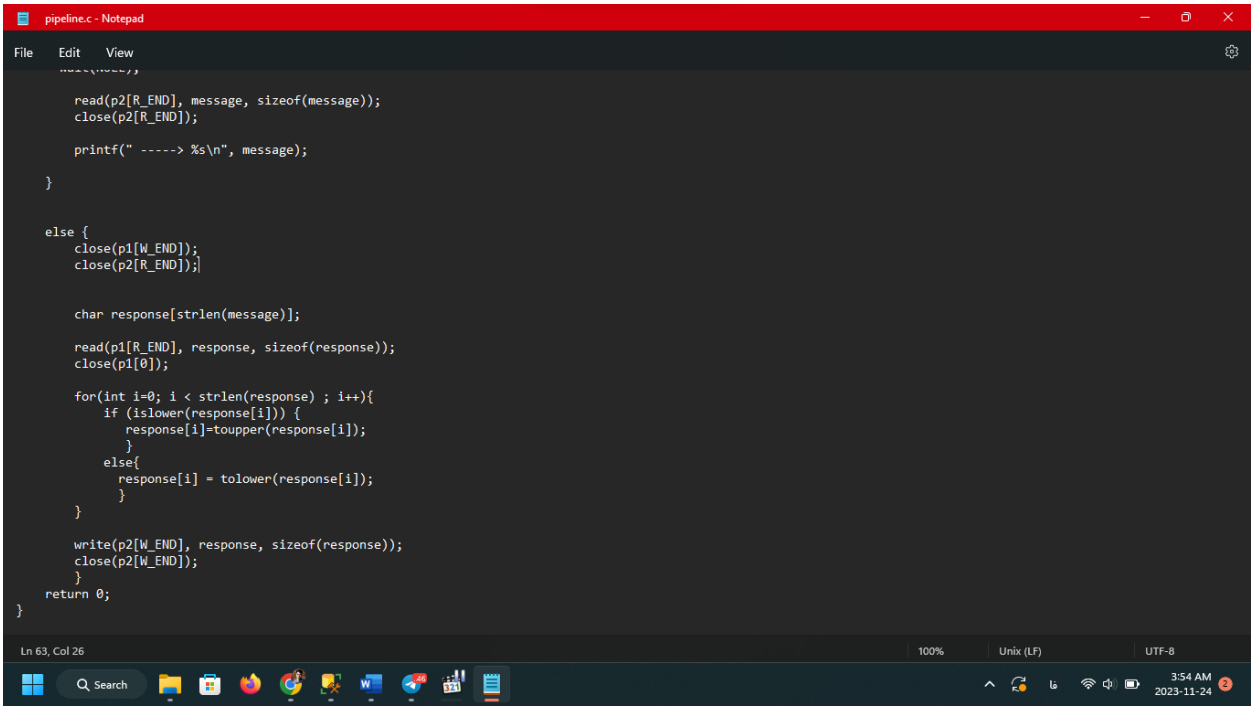
(HW3

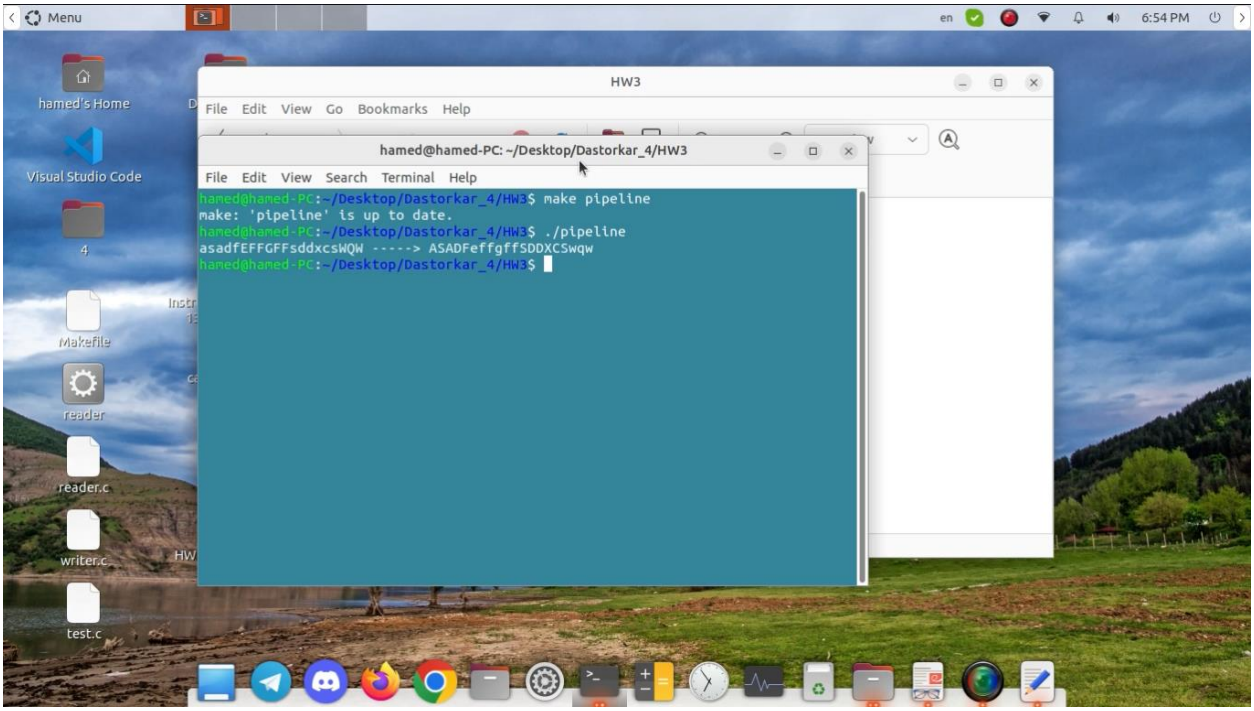
در این بخش که بخش پایانی است، از ما میخواهد که محیطی فراهم کنیم که در آن دو فرآیند با استفاده از خط لوله به تبادل یک پیام متنی بپردازند. فرآیند اول یک پیام متنی داراي حروف بزرگ و کوچک به فرآیند دوم ارسال می کند ، فرآیند دوم این پیام را دریافت می کند و حروف بزرگ را به حروف کوچک و حروف کوچک را به حروف بزرگ تبدیل می کند. برای ساخت این برنامه ما نیاز به ایجاد یک فایل c داریم که هردو فرایند را در آن ایجاد کرده و با استفاده از ارتباط خط لوله، نوشته را به نتیجه برسانیم.
برای ایجاد فرایند دوم ما از متد child استفاده میکنیم و آن را با استفاده از متد ()fork ایجاد میکنیم.
همچنین در ابتده فرایند پدر اول انجام میشود پس او باید بنویسد و باتوجه به خط لوله، فرزند باید بخواند.
به همین دلیل باید در ابتدا خواندن پدر و نوشتن فرزند را close کرد; سپس مقدار را نوشت و پس از آن نوشتن پدر را نیز قطع کرد. پس از انجام این مراحل، باید متد ()wait صدا زده شود تا عملیات خواندن فرزند انجام شود و سپس آن را ببندیم.

در بخش دوم، حال فرزند باید چیزی نوشته و پدر نیز بخواند. به همین دلیل باید نوشتن پدر و خواندن فرزند را نیز بست... و در بین کارها باید با استفاده از قابلیت برنامه c حروف نوشته هارا از بزرگ به کوچک و کوچک را به بزرگ تبدیل کرد.
حال عکس اجرای کد و همینطورکد برنامه در پایین آمده است:




pipeline.c - Notepad
File Edit View
pid = fork();
if(pid == -1){
perror("fork error");
exit(EXIT_FAILURE);
}
if (pid != 0) {
close(p1[R_END]);
close(p2[W_END]);
printf("%s", message);
write(p1[W_END], message, sizeof(message));
close(p1[W_END]);
wait(NULL);
read(p2[R_END], message, sizeof(message));
close(p2[R_END]);
printf(" -----> %s\n", message);
}
else {
close(p1[W_END]);
close(p2[R_END]);
Ln 33, Col 30
100%
Unix (LF)
UTF-8
Search
3:54 AM
2023-11-24


pipeline.c - Notepad
File Edit View
read(p2[R_END], message, sizeof(message));
close(p2[R_END]);
printf(" -----> %s\n", message);
}
else {
close(p1[W_END]);
close(p2[R_END]);
char response[strlen(message)];
read(p1[R_END], response, sizeof(response));
close(p1[0]);
for(int i=0; i < strlen(response) ; i++){
if (islower(response[i])) {
response[i]=toupper(response[i]);
}
else{
response[i] = tolower(response[i]);
}
}
write(p2[W_END], response, sizeof(response));
close(p2[W_END]);
}
return 0;
}
Ln 63, Col 26
100%
Unix (LF)
UTF-8
Search
3:54 AM
2023-11-24


Menu
en
6:54 PM
hamed's Home
Visual Studio Code
4
Makefile
reader
reader.c
writer.c
test.c
HW3
File Edit View Go Bookmarks Help
hamed@hamed-PC: ~/Desktop/Dastorkar_4/HW3
File Edit View Search Terminal Help
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW3$ make pipeline
make: 'pipeline' is up to date.
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW3$ ./pipeline
asadfEFFGFFsddxcsWQW -----> ASADFeffgffSDDXCSwqw
hamed@hamed-PC:~/Desktop/Dastorkar_4/HW3$