OS LAB WEEK6

Name: Sagnik Chatterjee

Roll No :61

Sec : B

REg: 180905478

Q1.

Codes:

q1\_reader.c

/\*

AUTHOR : SAGNIK CHATTERJEE

DATE : DEC 15,2020

USAGE : ./q1r

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

#define max 256

struct msg\_buffer {

long mesg\_type;

char mesg\_text[100];

} message;

int reverseDigits(int num)

{

int rev\_num = 0;

while (num > 0) {

rev\_num = rev\_num \* 10 + num % 10;

num = num / 10;

}

return rev\_num;

}

int isPalindrome(int n)

{

int rev\_num = reverseDigits(n);

if (rev\_num == n)

return 1;

else

return 0;

}

int main() {

key\_t key;

int msgid;

//ftok to generate unique key

key = ftok("progfile", 65);

//msgget creates a message queue

//and returns indentifer

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

//msgrev to receive message

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

//check if the messagedata is pallindrome or not

int number = atoi(message.mesg\_text);

printf("[STATUS] Data Received is :%d\n", number);

if (isPalindrome(number)==1) {

//yes this is a palindrome

printf("[STATUS] %d is a palindrome number.\n",number);

}

else {

printf("[STATUS] No this is not a palindrome number.\n");

}

//destroy the message queue

msgctl(msgid, IPC\_RMID, NULL);

return 0;

}

q1\_writer.c

/\*

AUTHOR : SAGNIK CHATTERJEE

DATE : DEC 15,2020

USAGE : ./q1w

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#define max 256

struct msg\_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("[STATUS] Enter the number to send to check for pallindrome : ");

fgets(message.mesg\_text, max, stdin);

// msgsnd to send message

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

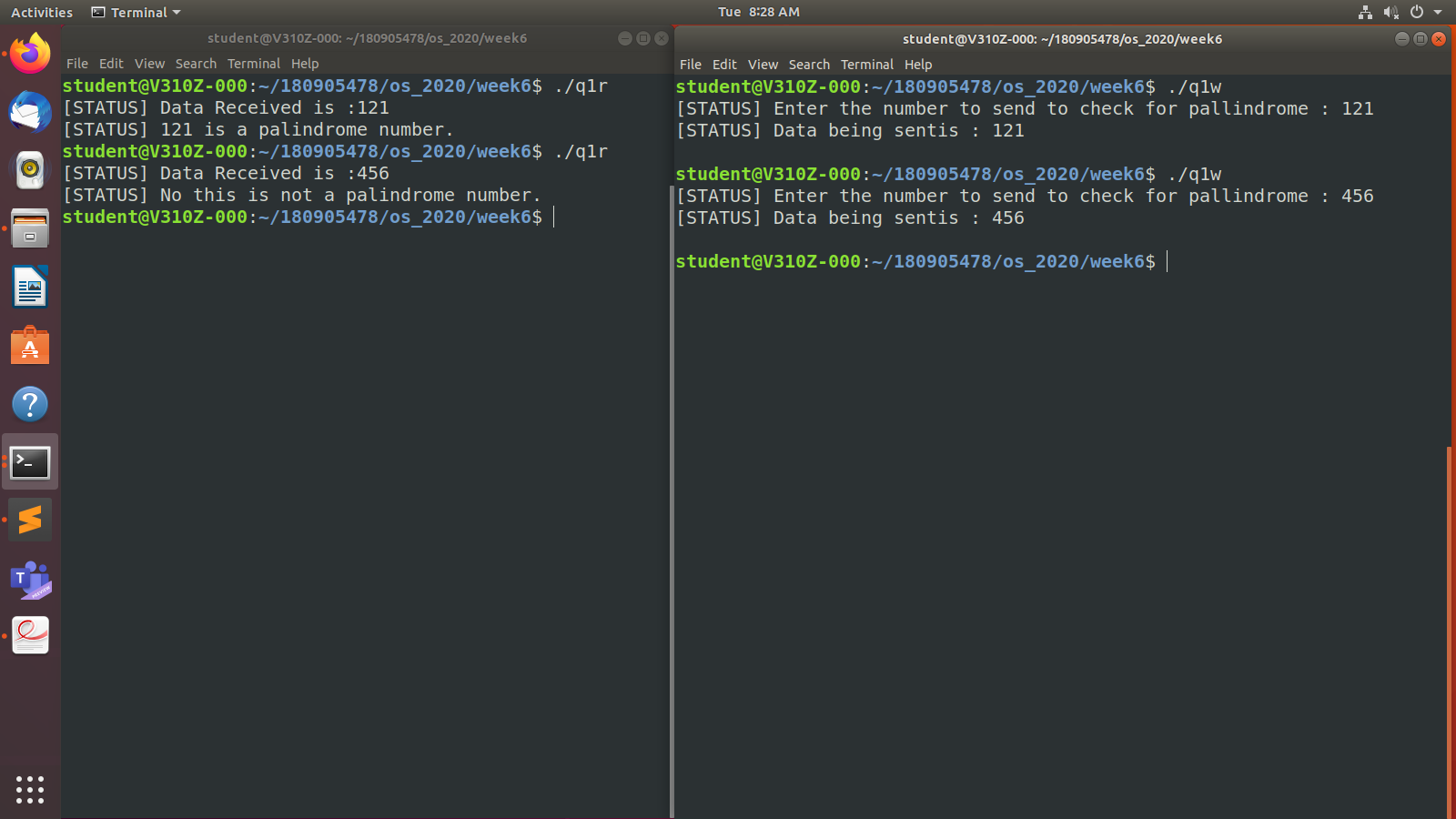
// display the message

printf("[STATUS] Data being sentis : %s \n", message.mesg\_text);

return 0;

}

Screenshot:



Q2

Code:

/\*

AUTHOR :SAGNIK CHATTERJEE

DATE : DEC 15,2020

USAGE : ./q2

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/shm.h>

#include <sys/stat.h>

#include <sys/wait.h>

#include <unistd.h>

#include <sys/types.h>

#define size 2 //because only allphabet sent

void nextalphabet(char \*alphabet){

char \*next= (char\*)calloc(2,sizeof(char));

next[1]='\0';

//handle corner cases of Z and z

if(\*alphabet=='Z'){

next[0]='a';

}

else if(\*alphabet=='z'){

next[0]='A';

}

//for other characters handle

else {

next[0]=\*alphabet+1;

}

\*alphabet=\*next;

}

int main(){

int segment\_id = shmget(IPC\_PRIVATE, size, S\_IRUSR | S\_IWUSR);

if(segment\_id<0){

printf("[ERROR] shmget error. \n");

exit(1);

}

char \*shared\_memory = (char \*)shmat(segment\_id, NULL, 0);

\*shared\_memory = '\0';

pid\_t pid;

int i;

\*shared\_memory = 'A';//from the first english alphabet

for (i = 0; i < 52; ++i) {

pid = fork();

if(pid<0){

printf("[ERROR] fork error \n");

exit(1);

}

if (pid == 0) {//for child process get the next alphabet

while (\*shared\_memory == '\0');

nextalphabet(shared\_memory);

exit(0);

} else {

//for parent process prints the next alphabet

//after the chracter has been incremented in the child process

printf("[STATUS] The next alphabet of %s is :- ", shared\_memory);

wait(NULL);

printf("%s\n", shared\_memory);

}

}

//close the shared\_memory

shmdt(shared\_memory);

}

Screenshot:

