**OS Lab**

**Session 3 – Lab 5 (15/12/2020)**

Parthivi Choubey CSE – B - 5th semester

180905456 Roll. no. - 60

**Question 1**

**Code**

Producer

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <assert.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <limits.h>

#include <fcntl.h>

#define FIFO\_NAME "/tmp/my\_fifo"

int main(int argc, char \*argv[])

{

int pipe\_fd;

int res;

int open\_mode = O\_WRONLY;

int bytes\_sent = 0;

int num;

if(access(FIFO\_NAME, F\_OK)==-1)

{

res = mkfifo(FIFO\_NAME, 0777);

if(res!=0)

{

fprintf(stderr, "Could not create fifo %s\n", FIFO\_NAME);

exit(EXIT\_FAILURE);

}

}

printf("Process %d opening write-only FIFO\n", getpid());

pipe\_fd = open(FIFO\_NAME, open\_mode);

if(pipe\_fd!=-1)

{

while(bytes\_sent < 4\*sizeof(int))

{

printf("Enter no. to be sent: ");

scanf("%d", &num);

res = write(pipe\_fd, &num, sizeof(int));

if(res==-1)

{

fprintf(stderr, "Write error on pipe\n");

exit(EXIT\_FAILURE);

}

bytes\_sent += res;

}

close(pipe\_fd);

}

else

{

perror("pipe");

exit(EXIT\_FAILURE);

}

printf("Process %d finished\nBytes sent = %d\n", getpid(), bytes\_sent);

exit(EXIT\_SUCCESS);

return 0;

}

Consumer

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <assert.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <limits.h>

#include <fcntl.h>

#define FIFO\_NAME "/tmp/my\_fifo"

int main()

{

int pipe\_fd;

int res;

int open\_mode = O\_RDONLY;

int bytes\_read = 0;

int buf=0;

printf("Process %d opening read-only FIFO\n", getpid());

pipe\_fd = open(FIFO\_NAME, open\_mode);

if(pipe\_fd!=-1)

{

res = read(pipe\_fd, &buf, sizeof(int));

while(res>0)

{

printf("%d\n", buf);

bytes\_read += res;

res = read(pipe\_fd, &buf, sizeof(int));

}

close(pipe\_fd);

}

else

{

perror("pipe");

exit(EXIT\_FAILURE);

}

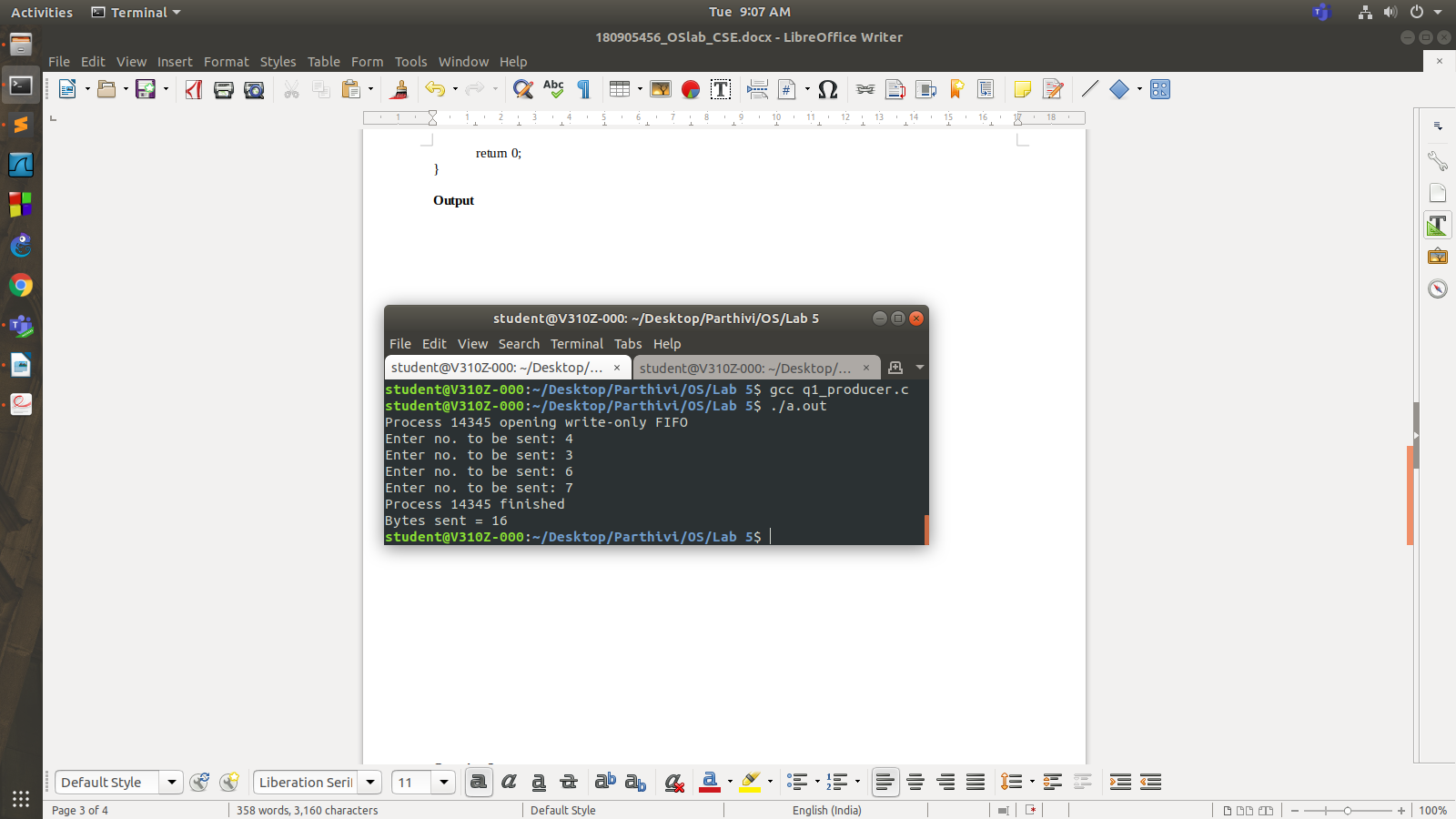
printf("Process %d finished\nBytes read = %d\n", getpid(), bytes\_read);

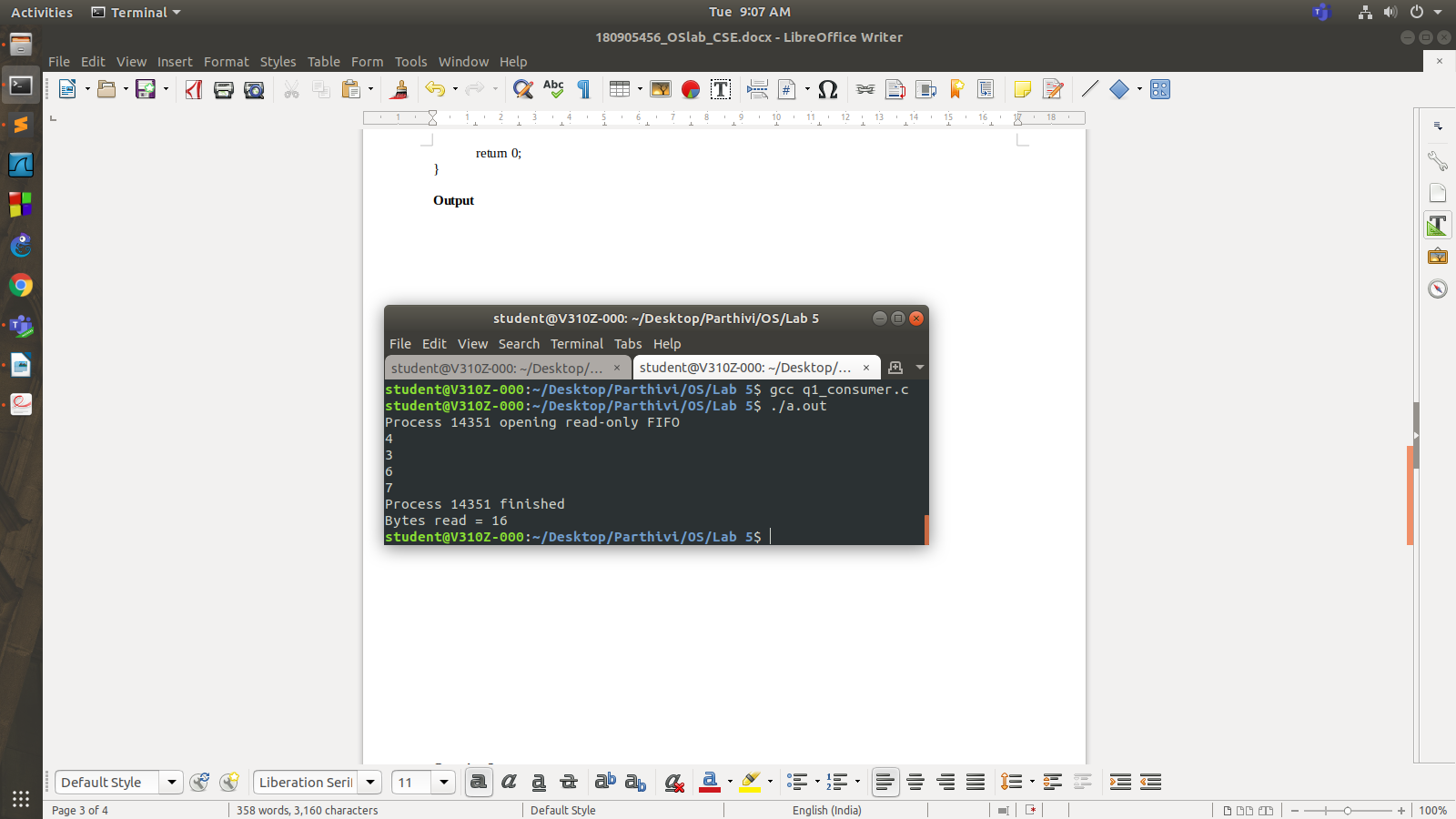
exit(EXIT\_SUCCESS);

return 0;

}

**Output**





**Question 2**

**Code**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <assert.h>

#include <sys/wait.h>

int main(int argc, char \*argv[])

{

int fd[2];

pid\_t pid;

char buf;

if(argc!=2)

{

printf("Invalid no. of arguments\n");

exit(EXIT\_FAILURE);

}

if(pipe(fd)==-1)

{

perror("pipe");

exit(EXIT\_FAILURE);

}

pid = fork();

if(pid==-1)

{

perror("fork");

exit(EXIT\_FAILURE);

}

else if(pid==0) //child process

{

printf("Reading in child...\n");

close(fd[1]); //close unused write end

while(read(fd[0], &buf, 1)>0)

write(STDOUT\_FILENO, &buf, 1);

write(STDOUT\_FILENO, "\n", 1);

close(fd[0]);

printf("Child ended\n");

exit(EXIT\_SUCCESS);

}

else //parent process

{

printf("Writing in parent...\n");

close(fd[0]); //close unused read end

write(fd[1], argv[1], strlen(argv[1]));

close(fd[1]); //reader will see EOF

wait(NULL); //wait for child to terminate

printf("Parent ended\n");

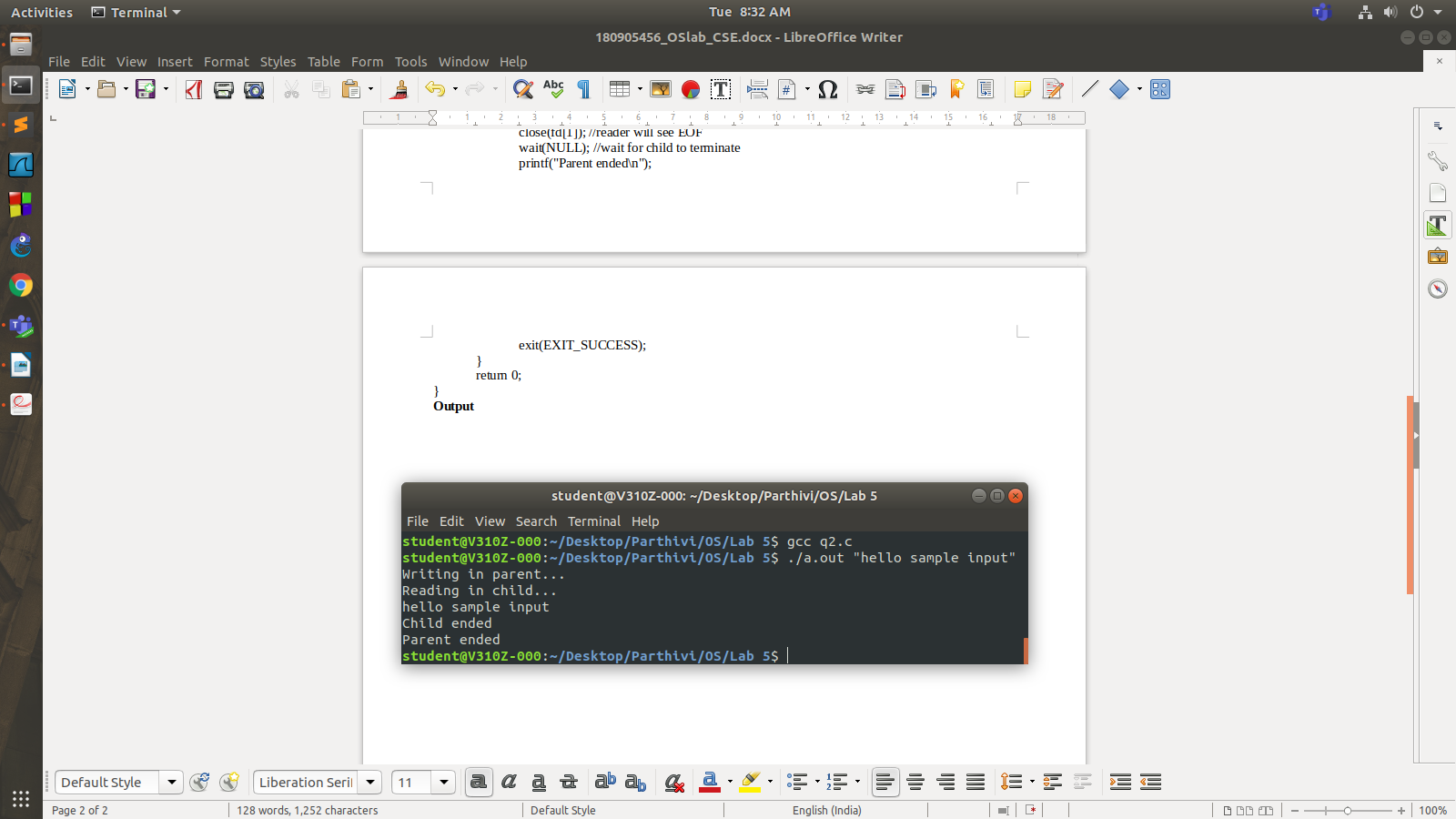
exit(EXIT\_SUCCESS);

}

return 0;

}

**Output**

****

**Question 3**

**Code**

Writer

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <assert.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <limits.h>

#include <fcntl.h>

#define FIFO\_NAME "/tmp/my\_fifo"

int main(int argc, char \*argv[])

{

int pipe\_fd;

int res;

int open\_mode = O\_WRONLY;

int num;

if(access(FIFO\_NAME, F\_OK)==-1)

{

res = mkfifo(FIFO\_NAME, 0777);

if(res!=0)

{

fprintf(stderr, "Could not create fifo %s\n", FIFO\_NAME);

exit(EXIT\_FAILURE);

}

printf("Fifo created\n");

}

printf("Process %d opening write-only FIFO\n", getpid());

pipe\_fd = open(FIFO\_NAME, open\_mode);

if(pipe\_fd!=-1)

{

printf("Enter no. to be sent: ");

scanf("%d", &num);

res = write(pipe\_fd, &num, sizeof(int));

if(res==-1)

{

fprintf(stderr, "Write error on pipe\n");

exit(EXIT\_FAILURE);

}

close(pipe\_fd);

}

else

{

perror("pipe");

exit(EXIT\_FAILURE);

}

printf("Process %d finished\nBytes sent = %d\n", getpid(), res);

exit(EXIT\_SUCCESS);

return 0;

}

Reader

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <assert.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <limits.h>

#include <fcntl.h>

#define FIFO\_NAME "/tmp/my\_fifo"

int main()

{

int pipe\_fd;

int res;

int open\_mode = O\_RDONLY;

int buf;

printf("Process %d opening read-only FIFO\n", getpid());

pipe\_fd = open(FIFO\_NAME, open\_mode);

if(pipe\_fd!=-1)

{

res = read(pipe\_fd, &buf, sizeof(int));

printf("%d\n", buf);

close(pipe\_fd);

}

else

{

perror("pipe");

exit(EXIT\_FAILURE);

}

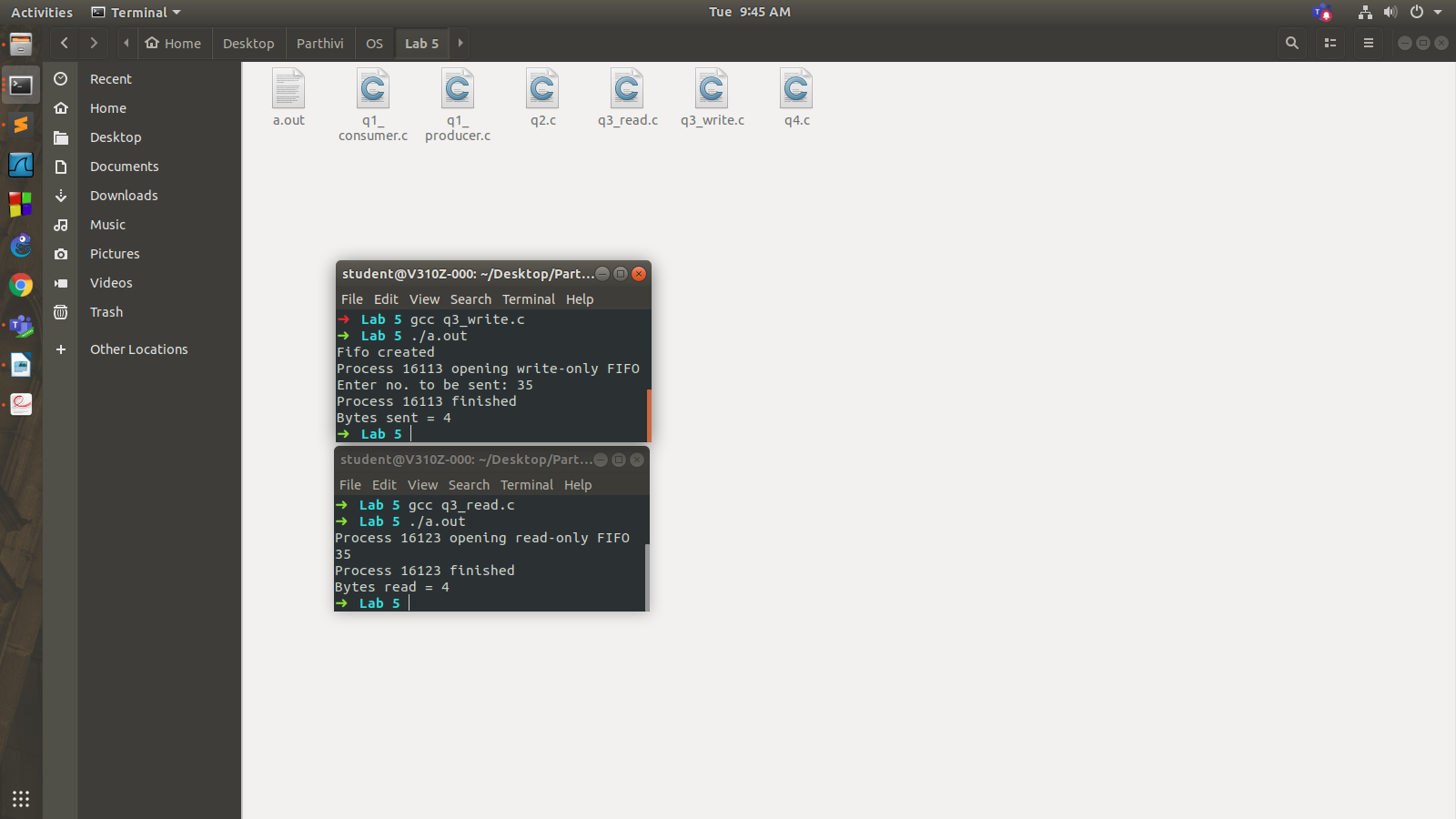
printf("Process %d finished\nBytes read = %d\n", getpid(), res);

exit(EXIT\_SUCCESS);

return 0;

}

**Output**

****

**Question 4 (read & write directly)**

**Code**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <assert.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <limits.h>

#include <fcntl.h>

int main(int argc, char \*argv[])

{

if(argc !=2)

{

printf("Invalid number of arguments\n");

exit(EXIT\_FAILURE);

}

char ch;

FILE \*fr, \*fw;

fr = fopen(argv[1], "rb");

if (fr == NULL)

printf("Input binary file can't be opened\n");

fw = fopen("q4.bin", "wb");

if (fw == NULL)

printf("Output binary file can't be opened\n");

while((ch = fgetc(fr)) != EOF)

putc(ch, fw);

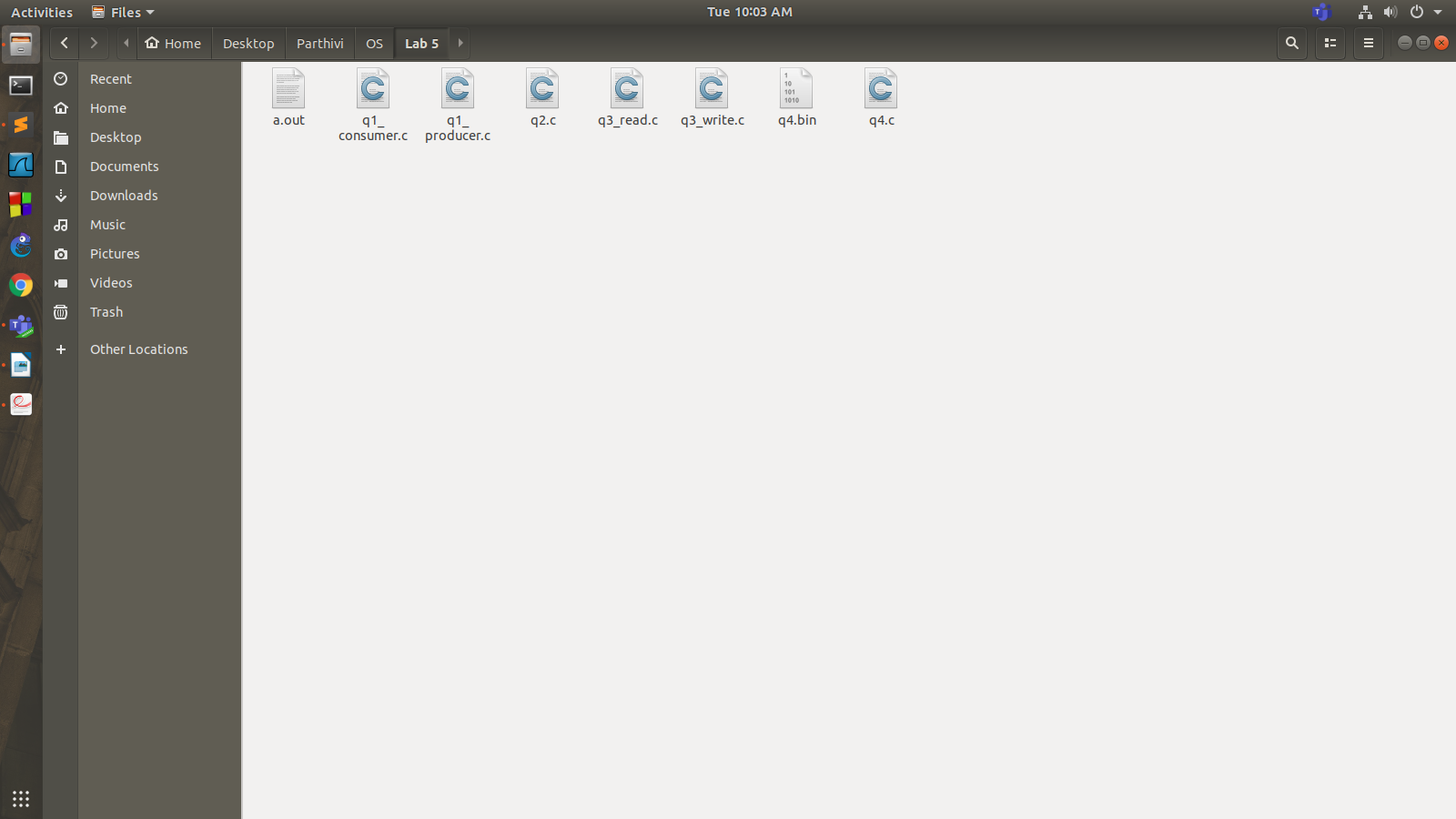
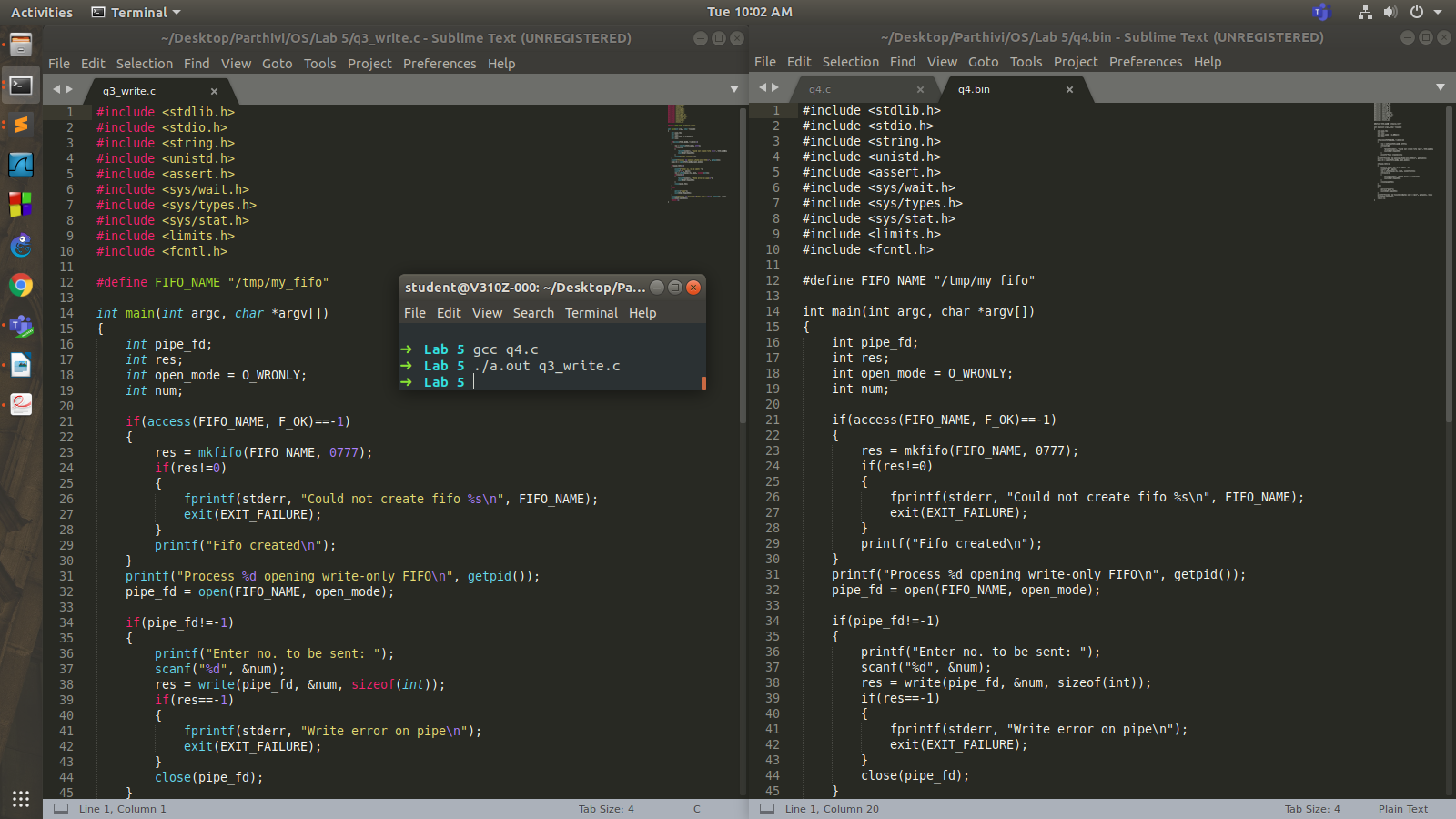
fclose(fr);

fclose(fw);

return 0;

}

**Output**

****

**Question 4 (via pipe)**

**Code**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <assert.h>

#include <sys/wait.h>

int main(int argc, char \*argv[])

{

int fd[2];

pid\_t pid;

char buf[1024];

if(argc!=2)

{

printf("Invalid no. of arguments\n");

exit(EXIT\_FAILURE);

}

if(pipe(fd)==-1)

{

perror("pipe");

exit(EXIT\_FAILURE);

}

pid = fork();

if(pid==-1)

{

perror("fork");

exit(EXIT\_FAILURE);

}

else if(pid==0) //child process

{

char ch;

FILE \*fw;

fw = fopen("q4.bin", "wb");

if (fw == NULL)

printf("Output binary file can't be opened\n");

printf("Reading in child...\n");

close(fd[1]); //close unused write end

while(read(fd[0], &buf, strlen(buf))>0)

{

fputs(buf, fw);

}

close(fd[0]);

printf("Child ended\n");

fclose(fw);

exit(EXIT\_SUCCESS);

}

else //parent process

{

char ch;

FILE \*fr = fopen(argv[1], "rb");

if (fr == NULL)

printf("Input binary file can't be opened\n");

printf("Writing in parent...\n");

close(fd[0]); //close unused read end

while(fgets(buf, 1024, fr) !=NULL)

{

write(fd[1], buf, strlen(buf));

}

close(fd[1]); //reader will see EOF

wait(NULL); //wait for child to terminate

printf("Parent ended\n");

fclose(fr);

exit(EXIT\_SUCCESS);

}

return 0;

}

**Output**

