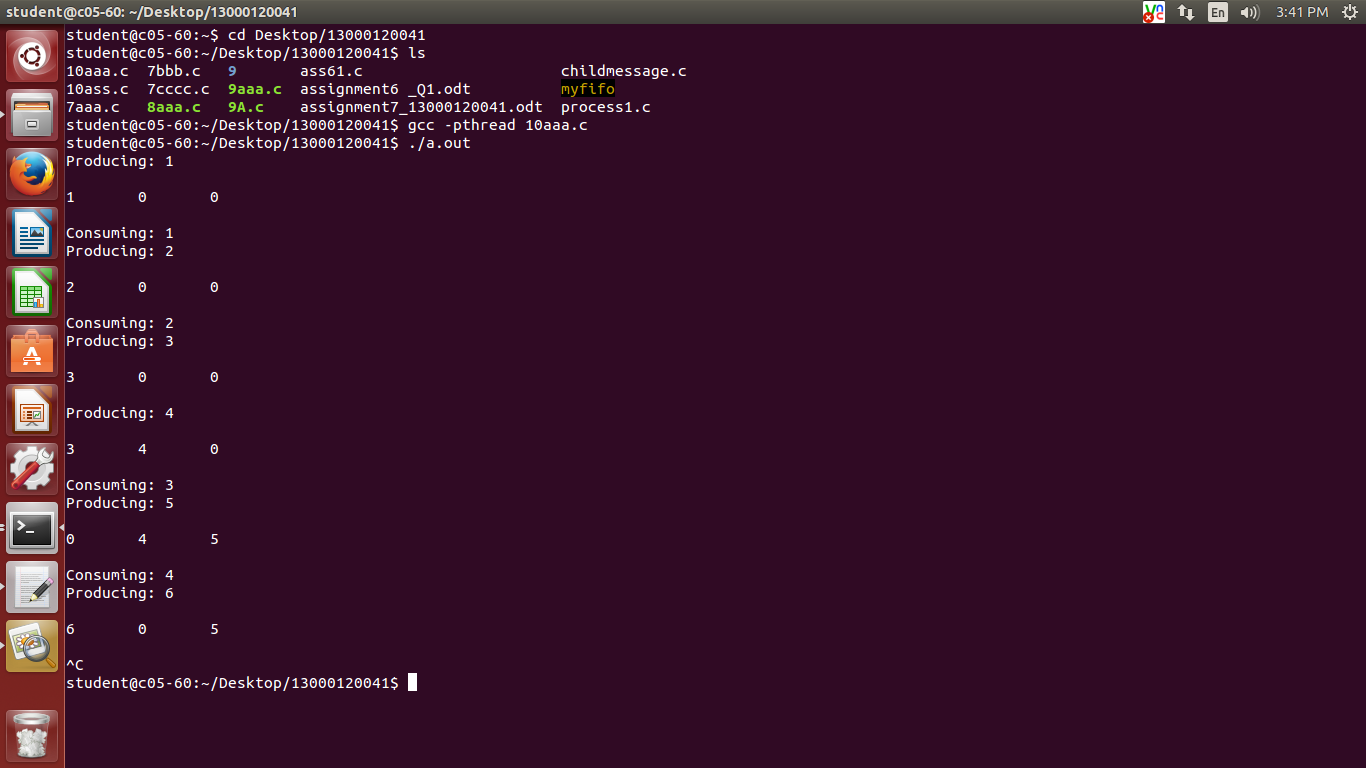
**Assignment 10**

**code**

#include<stdio.h>  
#include<stdlib.h>  
#include<sys/types.h>  
#include<unistd.h>  
#include<sys/wait.h>  
#include<semaphore.h>  
#include<pthread.h>  
#define SIZE 3  
sem\_t full, empty, mutex;  
int items[SIZE];  
int front = -1;  
int rear = -1;  
int x = 1;  
int isFull(){  
if((front == rear + 1) || (front == 0 && rear == SIZE - 1))  
return 1;  
return 0;  
}  
int isEmpty(){  
if(front == -1)  
return 1;  
return 0;  
}  
int enQueue(int element){  
if(isFull()){  
printf("\nQUEUE is full.\n");  
return -1;  
}  
else{  
if(front == -1)  
front = 0;  
rear = (rear + 1) % SIZE;  
items[rear] = element;  
return 0;  
}  
}  
int deQueue(){  
int element;  
int temp;  
if(isEmpty()){  
printf("\nQUEUE is empty.\n");  
return -1;  
}  
else{  
element = items[front];  
temp=element;  
if(front == rear){  
items[front]=0;  
front = -1;  
rear = -1;  
}  
else{  
items[front]=0;  
front = (front + 1) % SIZE;  
}  
return temp;  
}  
}  
void\* prod(void\* args){  
do{  
sleep(2);  
//printf("Producing: %d\n", x);  
sem\_wait(&empty);  
sem\_wait(&mutex);  
//printf("Producing: %d\n", x);  
if(enQueue(x) == 0)  
printf("Producing: %d\n", x);  
x++;  
int i;  
printf("\n");  
for(i = 0; i < SIZE; i++)  
printf("%d\t",items[i]);  
printf("\n\n");  
sem\_post(&full);  
sem\_post(&mutex);  
}  
while(1);  
}  
void\* con(void\* args){  
do{  
sleep(3);  
sem\_wait(&full);  
sem\_wait(&mutex);  
int y = deQueue();  
sem\_post(&empty);  
sem\_post(&mutex);  
printf("Consuming: %d\n", y);  
}  
while(1);  
}  
int main(){  
pthread\_t pd, cn;  
sem\_init(&empty, 0, 10);  
sem\_init(&full, 0, 0);  
sem\_init(&mutex, 0, 1);  
pthread\_create(&pd,NULL, &prod, NULL);  
pthread\_create(&cn,NULL, &con, NULL);  
pthread\_join(pd, NULL);  
pthread\_join(cn, NULL);  
return 0;  
}

**Output ss**

****