#include <unistd.h>

#include <string.h>

#include <stdio.h>

#define BUFSIZE 100

#define READ\_END 0

#define WRITE\_END 1

int main(){

    int pd[2]; // two pipe descriptors:pd[0] for reading and pd[1] for writing

    char buffer[BUFSIZ + 1];

    int returnstatus;

    returnstatus = pipe(pd);

    if (returnstatus == -1){

        printf("Unable to create pipe\n");

        return 1;

    }

    pid\_t fork\_result; fork\_result = fork();

    if (fork\_result == 0){ // child process receives data from parent

        read(pd[READ\_END], buffer, BUFSIZ);

        printf("\nChild Received: %s from the parent.\n", buffer);

        close(pd[READ\_END]);

    } else { // parent process pass string to child

        char some\_data[10];

        printf("\nEnter some data to be passed to child process:");

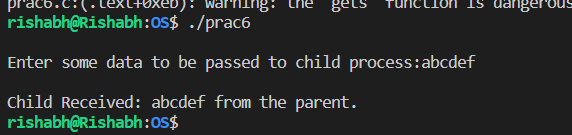
        gets(some\_data); write(pd[WRITE\_END], some\_data, strlen(some\_data));

        close(pd[WRITE\_END]);

    }

    return 0;

}



#include<stdio.h>

#include<stdlib.h>

int mutex=1,full=0,empty=3,x=0;

int main(){

int n; void producer(); void consumer();

int wait(int); int signal(int);

printf("\n1.Producer\n2.Consumer\n3.Exit");

while(1){

printf("\nEnter your choice:");

scanf("%d",&n);

switch(n){

case 1: if((mutex==1)&&(empty!=0))

producer();

else

printf("Buffer is full!!");

break;

case 2: if((mutex==1)&&(full!=0))

consumer();

else

printf("Buffer is empty!!");

break;

case 3:

exit(0);

break;

}

}

return 0;

}

int wait(int s){

return (--s);

}

int signal(int s){

return(++s);

}

void producer(){

mutex=wait(mutex);

full=signal(full);

empty=wait(empty);

x++;

printf("\nProducer produces the item %d",x);

mutex=signal(mutex);

}

void consumer(){

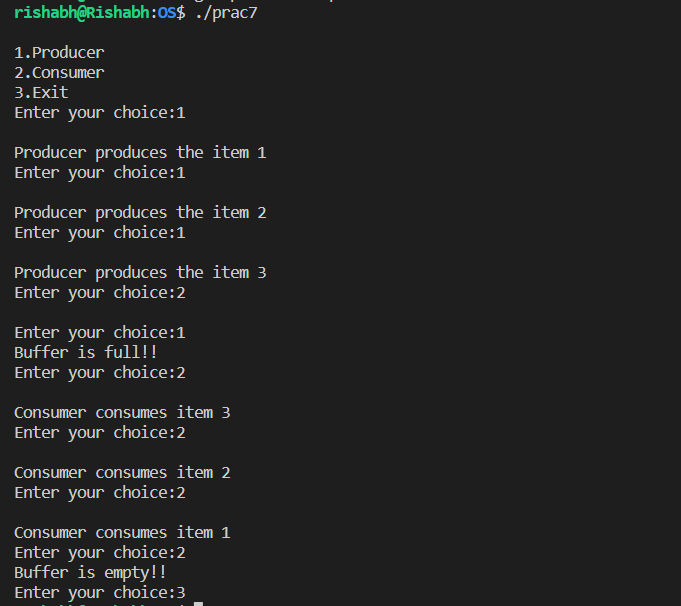
mutex=wait(mutex);

full=wait(full);

empty=signal(empty);

printf("\nConsumer consumes item %d",x);

x--;

 mutex=signal(mutex); }