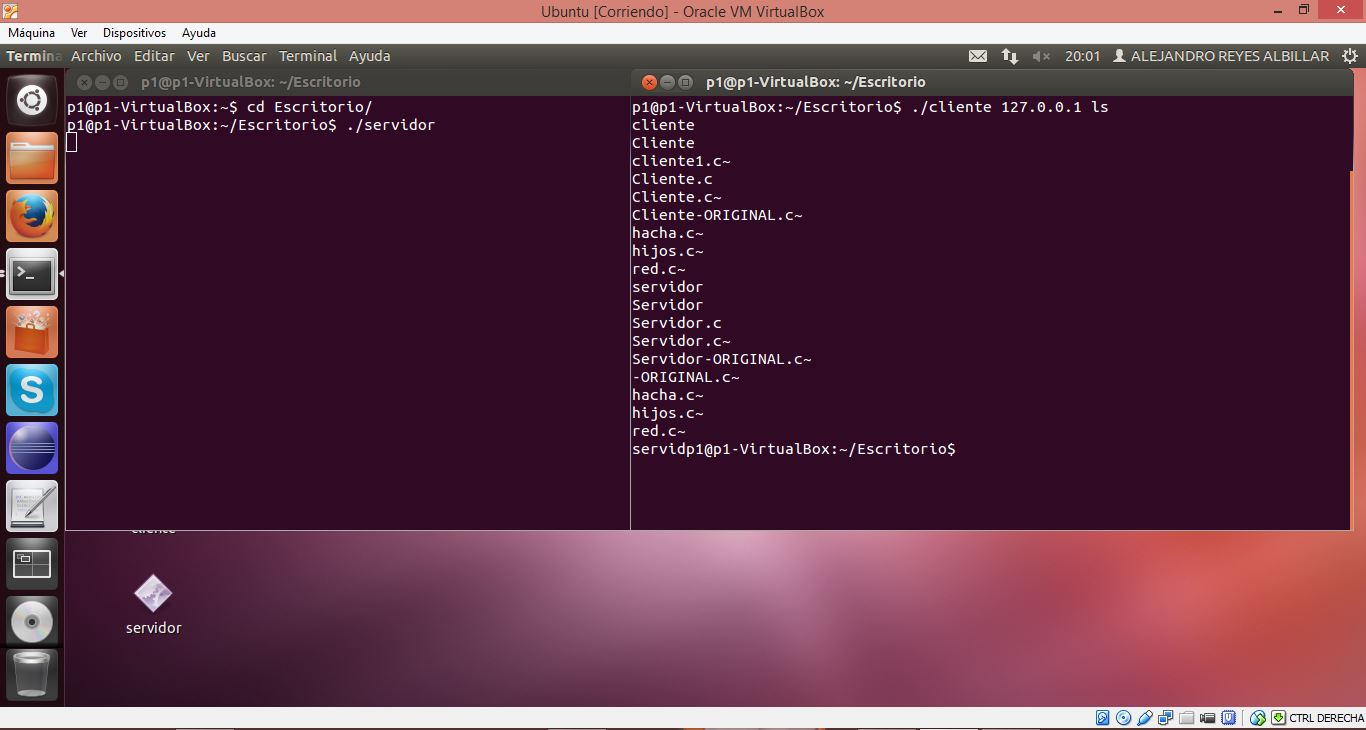
Práctica 2: Comunicaciones en red

1º) Ejercicio Cliente- Servidor:

Tras múltiples intentos de que funcionara el código, tanto del cliente como del servidor, de manera infructuosa decidí preguntarle a compañeros acerca de cómo lo habían hecho ellos la práctica y si me enseñaban su código para poder ver lo que fallaba en el mío. Tras encontrar el error del mío, una llamada a fork a la que le faltaban los paréntesis, compilé el código y ejecuté tanto servidor como cliente, los dos con el mismo resultado, error. El código que se adjunta a la práctica es el que no funciona, compila pero no funciona. Personalmente he comparado el código con el del profesor y el de distintos compañeros y no se ve diferencia alguna a parte del orden en las declaraciones iniciadas y el nombre de algunas variables, por el resto el código es idéntico. Sin embargo, si ejecuto el código de otros compañeros después de haberlo compilado en el mismo equipo y bajo las mismas circunstancias (mismo lenguaje y compilador), el programa del compañero funciona correctamente, conecta y ejecuta las instrucciones correctamente como se muestra en la siguiente imagen.



A continuación se indican tanto el código del cliente como el del servidor escritos por mí, aunque también están en los archivos .c que se encuentran en el .zip junto a esta memoria.

Servidor:

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#define PUERTO 9999

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

char buffer[256];

int tam, fd, fd2;

struct sockaddr\_in servidor;

struct sockaddr\_in cliente;

servidor.sin\_family=AF\_INET;

servidor.sin\_port=htons(PUERTO);

servidor.sin\_addr.s\_addr=INADDR\_ANY;

if(fd=socket(AF\_INET,SOCK\_STREAM,0)==-1){

perror("Error en el socket.\n");

exit(-1);

}

else{

printf("El socket funciona bien.\n");

if(bind(fd, (struct sockaddr\*) &servidor, sizeof(struct sockaddr))==-1){

perror("Error en el bind.\n");

exit(-1);

}

else{

printf("El bind funciona bien.\n");

if(listen(fd, 5)==-1){

perror("Error en el listen.\n");

}

else{

printf("El listen funciona bien.\n");

for(;;){

tam=sizeof(struct sockaddr\_in);

//Recibe de connect

if(fd2=accept(fd, (struct sockaddr\*) &cliente, &tam)==-1){

perror("Fallo en el accept.\n");

exit(-1);

}

else{

printf("Cliente aceptado.\n");

if(fork()==0){//Si es el hijo

close(fd);

while(read(fd2, buffer, sizeof(buffer))>0){

close(1);

dup(fd2);

execlp(buffer, buffer, NULL);

write(fd2, execlp(buffer, buffer, NULL),100);

}

close(fd2);

exit(0);

}

else{

close(fd2);

}

}

}

}

}

}

}

Cliente:

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#define PUERTO 9999

#define MAX\_BUFFER 10000

struct sockaddr\_in cliente;

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

char buffer[MAX\_BUFFER];

int fin, fd, fd2, tam; //File Descriptor

if(argc<3 || argv[2]==NULL){

perror("Error de argumentos en el cliente. Se ha de llamar de este modo: ./Cliente <ipServidor> <Comando>\n");

exit(-1);

}

//AF\_INET: Address Family Internet, SOCK\_(STREAM|DEGRAM):En el protocolo de transporte utilizarÃ¡ TCP si ponemos stream,o UDP si utiliza degram, que significa datagrama

cliente.sin\_family=AF\_INET;

cliente.sin\_port=htons(PUERTO);//Puertos por debajo de 1023 son puertos administrativos por lo que necesitarÃ¡n permisos de superusuario. Empleamos htons: host to network short y se le pasa el puerto

cliente.sin\_addr.s\_addr=inet\_addr(argv[1]);

if(fd=socket(AF\_INET,SOCK\_STREAM,0)==-1){

perror("Error en el socket del cliente.\n");

exit(-1);

}

else{

printf("Socket abierto.\n");

//Solicitud a accept

if((connect(fd, (struct sockaddr\*) &cliente, sizeof(struct sockaddr)))==-1){//Se le pasa el descriptor de socket, se hace un cast de servidor y se le pasa el tamaÃ±o del tipo

perror("Error al conectar con el servidor.\n");

exit(-1);

}

else{

printf("Conexión establecida con servidor.\n");

write(fd,argv[2],100);

while(read(fd,buffer, sizeof(buffer))>0){

printf("%s", buffer);

}

}

}

}

2º) Ejercicio semáforos:

El ejercicio de semáforos lo hemos realizado en el entorno JBACI, un entorno especialmente diseñado para esta actividad que nos permite visualizar gráficamente las acciones de los semáforos en cuestión.

A continuación se muestra el código que se encuentra en la carpeta correspondiente del zip que contiene este documento:

//Recursos compartidos

#include "gdefs.cm"

semaphore m; //Semaforo que controla la creación de piezas de la máquina

semaphore p; //Semáforo que controla el acceso a las piezas

semaphore r; // Semáforo que controla al robot

int piezas; //Nº de piezas creadas

const int N=3; //Total de piezas que pueden haber en la cinta

int piezascogidas; //Total de piezas recogidas

int paquete; //Nº de paquetes creados que contienen 3 piezas cada uno

int oper; //Nº de veces que ha pasado el operario

const int BG=20; //Fondo Blanco de los gráficos

int CIN=21; //Cinta transportadora

int P1=22; //Pieza 1

int P2=23; //Pieza 2

int P3=24; //Pieza 3

int ROBASE=25; //Base del Robot

int ROB=26; //Robot

int CABOP=27; //Cabeza del operario

int CUEOP=28; //Cuerpo del operario

int MAQ=29; //Máquina

void maquina(){

while(1){

wait(m);

wait(p);

if(piezas<N){//Antes de nada visualizar

if(piezas==0){

makevisible(P1,0);

makevisible(P2,0);

makevisible(P3,0);

}

if(piezas==1){

makevisible(P3,1);

}

if(piezas==2){

makevisible(P2,1);

makevisible(P3,1);

}

if(piezas==3){

makevisible(P1,1);

makevisible(P2,1);

makevisible(P3,1);

}

piezas++; //Pone una nueva pieza en la cinta

cout<<"Soy la máquina y añado una pieza. Hay "<< piezas << " piezas."<<endl;

//Se mueven las piezas desde donde se crean hasta el lugar que les corresponde

if(piezas==1){

moveto(MAQ,100,150);

changecolor(MAQ, BLUE);

makevisible(P1,1);

changecolor(MAQ, YELLOW);

moveto(MAQ,100,100);

makevisible(P1,0);

makevisible(P2,1);

makevisible(P2,0);

makevisible(P3,1);

}

if(piezas==2){

moveto(MAQ,100,150);

changecolor(MAQ, BLUE);

makevisible(P1,1);

changecolor(MAQ, YELLOW);

moveto(MAQ,100,100);

makevisible(P1,0);

makevisible(P2,1);

}

if(piezas==3){

moveto(MAQ,100,150);

changecolor(MAQ, BLUE);

makevisible(P1,1);

changecolor(MAQ, YELLOW);

moveto(MAQ,100,100);

}

}

signal(p);

signal(r);

}

}

void robot(){

while(1){

wait(r);

wait(p);

piezas--; //Quita una pieza de la cinta

piezascogidas++; //Pone una pieza en un paquete

if(piezas<0){piezas=0;} //Para evitar errores de las piezas de más quitadas(piezas<0) //Una vez quitada la pieza quitarla y mover el resto a sus respectivas posiciones

if(piezas==0){

changecolor(ROB, GREEN);

moveto(ROB,500,100);

makevisible(P3,0);

moveto(ROB,500,50);

changecolor(ROB, RED);

}

if(piezas==1){

changecolor(ROB, GREEN);

moveto(ROB,500,100);

makevisible(P3,0);

moveto(ROB,500,50);

changecolor(ROB, RED);

makevisible(P2,0);

makevisible(P3,1);

}

if(piezas==2){

changecolor(ROB, GREEN);

moveto(ROB,500,100);

makevisible(P3,0);

moveto(ROB,500,50);

changecolor(ROB, RED);

makevisible(P2,0);

makevisible(P3,1);

makevisible(P1,0);

makevisible(P2,1);

}

cout << "Soy el robot y cojo una pieza para meterla a un paquete. Hay " << paquete << " paquetes." << endl;

signal(p);

signal(m);

if(piezascogidas==3){ //Cuando Se han cogido 3 piezas se completa un paquete

paquete++;

piezascogidas=0;

}

}

}

void operario(){

while(1){

wait(p);

if(piezas>=3){

piezas=piezas-3;

signal(m);signal(m);signal(m);

if(piezas==0){

moveto(CABOP,275,300);

moveto(CUEOP,300,350);

changecolor(CABOP,GREEN);

changecolor(CUEOP,GREEN);

makevisible(P1,0);

makevisible(P2,0);

makevisible(P3,0);

changecolor(CUEOP,RED);

changecolor(CABOP,RED);

moveto(CUEOP,300,400);

moveto(CABOP,275,350);

}

cout<<"Soy el operario y cojo 3 piezas para control de calidad"<<endl;

oper++;

if(oper==10){exit(-1);} // Cuando el operario haya pasado 10 veces el programa parará.

}

signal(p);

}

}

void main(){

create(BG,RECTANGLE, WHITE,0,0,600,450); //Fondo de Pantalla

create(CIN, RECTANGLE, BLACK, 0,150,600,150); //Cinta transportadora

create(ROBASE, RECTANGLE, BLUE, 450,50,100,50); //Base del robot

create(ROB, TRIANGLE, RED, 500,50,100,50); //Cabeza del Robot

create(MAQ, TRIANGLE, YELLOW, 100,100,50,-50); //Maquina

create(CABOP,CIRCLE,RED,275,350,50,50); //Cabeza del operario

create(CUEOP,TRIANGLE,RED,300,400,50,50); //Cuerpo del operario

create(P1, RECTANGLE,MAGENTA,50,200,100,50); //Pieza 1

create(P2, RECTANGLE,MAGENTA,250,200,100,50); //Pieza 2

create(P3, RECTANGLE,MAGENTA,450,200,100,50); //Pieza 3

initialsem(m, N); //Se inicializa a la máquina con la constante N se bloqueará si es -1

initialsem(p,1); //El semáforo de acceso a piezas se inicializa a 1

initialsem(r,0); //El semáforo del robot se inicializa a 0 para que se bloquee en el caso que sea -1

piezas=0; //Inicializamos las piezas a 0

piezascogidas=0; //Inicializamos las piezas que cogemos para un paquete a 0

paquete=0; //Inicializamos los paquetes de la producción a 0

oper=0; //Inicializamos las veces que un operario pasa revisión de piezas a 0

cobegin{

maquina();

robot();

operario();

}

}

Para poder visualizar los gráficos en la ventana TheGraphics de JBACI hemos de incluir las librerías correspondientes y utilizar las opciones indicadas por los pdf’s de ayuda proporcionados por el profesor.

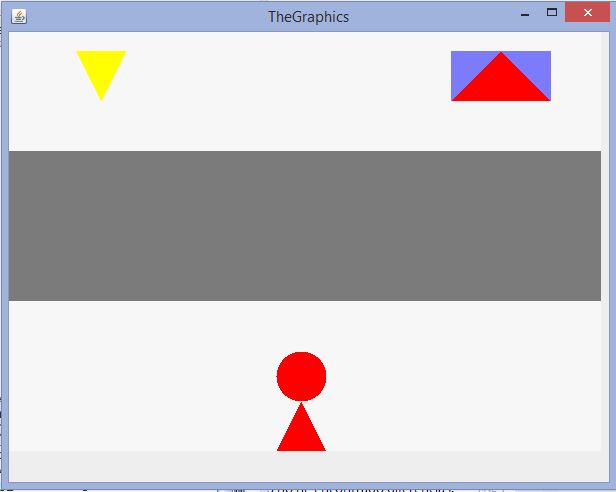
Se utilizan 3 semáforos principales que son m, p y r referentes a la máquina, las piezas y el robot respectivamente.

El semáforo m se inicializa a un número de piezas, en mi caso 3, que es el que puede haber sobre la cinta transportadora, y cada vez que se llama a maquina() se disminuye en 1, accediendo a las piezas con un semáforo mutex, denominado p e inicializado a 1, para que únicamente pudiera acceder un módulo al mismo tiempo. Cuando se añade a la cinta una pieza se hace un signal(r) para que el robot, cuyo semáforo se inicializa a 0 pueda ser accedido.

El módulo robot recoge una pieza de la cinta, haciendo wait(r), accediendo a las piezas haciendo wait(p) y signal(p). Las piezas recogidas se introducen en paquetes en los que entran 3 piezas. Por lo tanto la variable paquetes aumenta en 1 por cada 3 piezas que son quitadas de la cinta. Para evitar errores se ha colocado una comprobación en la que, si las piezas llegan a ser menor que 0 se reinicializa piezas a 0.

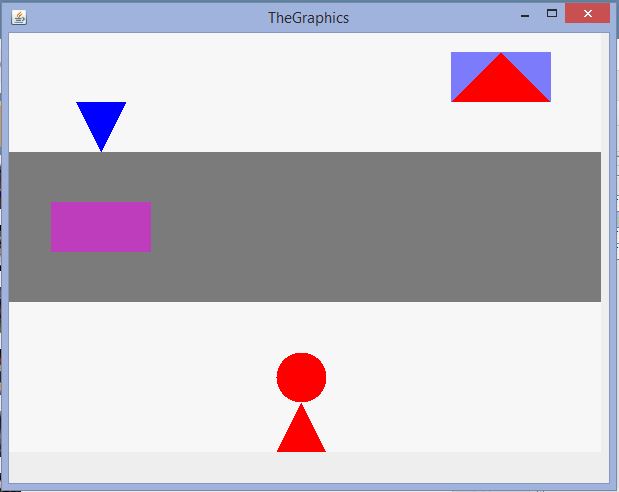
Cada cierto tiempo, y cuando hay tres piezas sobre la cinta transportadora, un operario retira las 3 piezas existentes para control de calidad y realiza 3 veces signal(m) para bloquear la máquina. Si quitas 3 piezas has de crear 3 piezas para evitar errores.

A continuación se muestran 4 imágenes en las que se muestran:

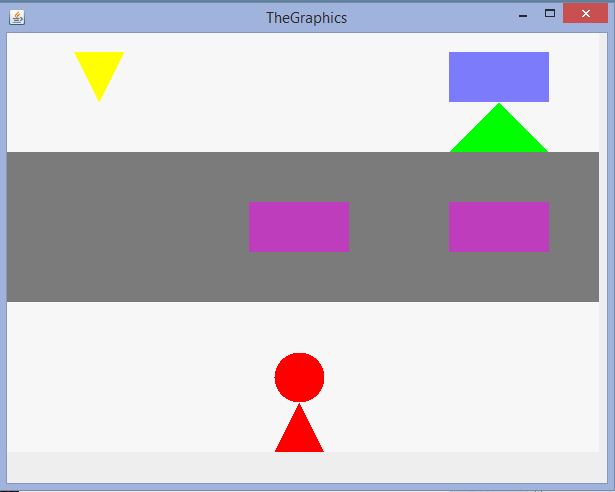


La situación inicial:

La situación en la que se crea una pieza:



La situación en la que el robot coge una pieza:



La situación en la que el operario recoge las 3 piezas:

