

# PRACTICA 2

CARLOS PRADOS SESMERO

DIEGO SÁNCHEZ MARCOS

4L

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

2016

## ÍNDICE

Planteamiento del problema.....	3
Soluciones tomadas para la solución del problema.....	3
Explicación de ciertos aspectos de la práctica.....	4
Código del programa.....	8
<i>Control.cpp</i> .....	8
<i>Depósito.cpp</i> .....	10
<i>Sensor.cpp</i> .....	12
<i>Válvula.cpp</i> .....	13
Ejemplo de ejecución .....	14
Bibliografía.....	15

## Planteamiento del problema

El problema al que nos enfrentamos consiste en controlar el nivel de un líquido en un depósito de forma que este permanezca siempre dentro de unos márgenes máximo y mínimo. El nivel del líquido en el depósito disminuye en 100 L/s cuando la válvula superior está cerrada. Para realizar el control se dispone de un sensor que envía información del nivel al sistema de control. Éste es el encargado de activar o desactivar el paso de agua a través de la válvula superior en base a la señal recibida. La apertura de esta válvula aporta 400L/s. Puesto que la salida del depósito está siempre abierta (decremento constante de 100L/s) el incremento de volumen de líquido neto es de 300L/s.

Siempre y cuando el nivel de líquido del depósito supere el nivel máximo se cerrará la válvula anteriormente descrita y siempre que el nivel sea inferior al nivel mínimo se abrirá la válvula. Para ello hemos desarrollado cuatro procesos: Control, Sensor, Válvula y Depósito.

## Soluciones tomadas para la solución del problema

Utilizaremos colas de mensajes y memoria compartida para su comunicación de modo que dichos procesos se relacionen de la siguiente manera:

- Disponemos de dos colas de mensajes, una que comunica el proceso de Control con el proceso de Válvula, la cual indica a la válvula si debe estar abierta o si debe estar cerrada, y una cola que comunica el proceso de Sensor y el proceso de Control, la cual indica el nivel del depósito que determina el sensor.
- Disponemos de dos memorias compartidas. Una de ellas comunica el proceso de Válvula con el proceso de Depósito de modo que se comunica al proceso Depósito el estado de la válvula. La otra comunica el proceso Depósito con el proceso Sensor indicando al sensor cual será el nivel actual.
- El proceso Control funciona de modo que recibe el nivel del depósito del proceso Sensor y procesa esta información de modo que si el nivel es menor que el nivel mínimo envía al proceso válvula que debe estar abierta, y en el caso de que el nivel sea mayor que el máximo se cierre la válvula. En el caso de que el nivel del depósito esté entre los niveles máximo y mínimo el estado de la válvula no cambiará.  
El proceso Control es el encargado de crear las dos colas, colaValve y colaSensor.
- El proceso Válvula es el encargado de abrir o cerrar la válvula dependiendo de la señal recibida desde Control.  
Este proceso abre la cola colaValve la memoria compartida memoValve.
- El proceso Depósito funciona de modo que recibe una señal del estado de la válvula y aumenta el nivel del depósito en el caso de estar abierta y le disminuye en el caso de que esté cerrada. Cabe destacar que el depósito inicialmente está a un nivel igual a 0.  
El proceso Depósito es el encargado de crear las dos memorias compartidas, memoValve y memoNivel (la cual la inicia a 0).
- El proceso Sensor tiene como objetivo detectar el nivel del depósito y enviar dicha información a Control.  
Este proceso abre la cola colaSensor y abre la memoria compartida memoNivel.

Para la parada de los procesos se realizará con el comando Ctrl-C e inmediatamente se procederá a borrar los ficheros que hayan quedado en los directorios /dev/shm y /dev/mqueue. Para ello hemos desarrollado dentro de nuestro programa un tratamiento de la señal Ctrl-C en el proceso de control para que envíe orden de finalización al resto de procesos y de esta manera se elimine los mecanismos de comunicación antes de la finalización.

Desde la línea de comandos podemos introducir los valores de nivel máximo y mínimo del depósito, de modo que al ejecutar el programa control podemos escribir dos valores que se corresponderán con el valor máximo y mínimo indistintamente del orden de introducción.

En nuestro programa si queremos “matar” el proceso debemos introducir el comando Ctrl-C en la línea de comandos, pero en el caso de que matemos el proceso de la Válvula o el proceso del Sensor hemos desarrollado un mecanismo de seguridad que provoca el salto de una alarma en el proceso de Control. Es decir, si matamos uno de los dos procesos nos saltará un error en el proceso de Control deteniendo dicho proceso, y junto a dicho proceso el resto puesto que están relacionados constantemente. Dicha alarma saltará en el caso de que el proceso Control no reciba información de Válvula o Sensor durante un cierto tiempo programable.

### **Explicación de ciertos aspectos de la práctica**

Durante la realización de la práctica nos hemos encontrado con ciertos aspectos que nos dificultan el correcto desarrollo, se procederá a explicar pormenorizadamente los problemas encontrados y las soluciones aportadas a ellos.

Problema: Las zonas de memoria compartida solo pueden contener caracteres tipo “char”.

Solución: Para pasar a formato “int” los caracteres de tipo “char” utilizamos la función “atoi” la cual nos realizará el objetivo deseado cuando sea necesario sumar o restar una cantidad determinada. Para devolverlo al formato “char” utilizaremos la función “sprintf” y en el caso de que sea necesario copiar dos cadenas de forma clara utilizaremos la función “strcpy”. Se muestra un ejemplo en las siguientes imágenes:

```
sprintf(p_aux, "%d", nivel);

nivel=atoi(cadenaRecibida);
if(nivel<nivelMin)
    strcpy(mandaValvula, "ON ");
if(nivel>nivelMax)
    strcpy(mandaValvula, "OFF");
```

Problema: No podemos trabajar con la memoria compartida como una variable típica de C++.

Solución: Declaramos un puntero que nos permitirá trabajar como una variable normal de C++.

```
cadena_valvula = (char *)estado_llave.getPointer();
p_aux=(char *)estado_deposito.getPointer();
```

Problema: Tratamiento de señales como “Ctrl-C” para terminar un proceso y que antes ejecute ciertas líneas de código.

Solución: Declaración de un manejador de señales que nos permitirá cambiar las acciones predeterminadas para distintos tipos de interrupciones, de esta forma nos aseguraremos que se borran correctamente todas las variables compartidas. Ejemplo de dicha solución:

```
void manejador(int signum)
{
    cout<<endl<<"Fin de programa"<<endl;

    s_deposito.close();
    s_sensor.close();

    estado_llave.cerrar();
    estado_deposito.cerrar();

    estado_llave.unlink();
    estado_deposito.unlink();

    s_deposito.unlink();
    s_sensor.unlink();

    exit(0);
}
```

Problema: Cerrar desde control Válvula y Sensor.

Solución: Al ejecutar Válvula o Sensor envían a control su pid (número de programa), si se produce un fallo en estos será necesario reasignar este pid.

Problema: Comprobación del estado de Válvula y Sensor, si cualquiera de estos dos falla Control quedaría bloqueado.

Solución: Lo realizaremos mediante Alarmas al procesador, estas envían una señal si la alarma no se desactiva en un tiempo fijado previamente, esta señal puede ser manejada como la anteriormente comentada.

```
signal(SIGALRM, alarma_sensor);
alarm(2);

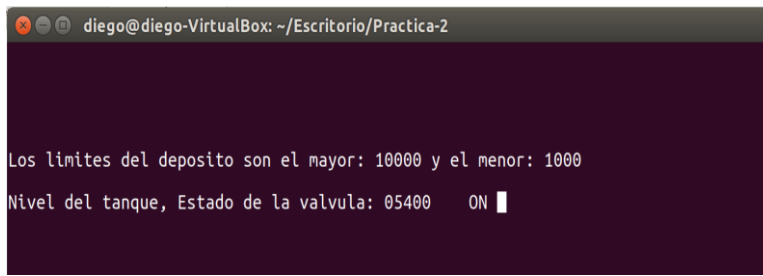
cola_sensor.receive(cadenaRecibida, TAMCADENA*sizeof(char));

if(atoi(cadenaRecibida)!=-1)
{
    alarm(0);
}

void alarma_sensor(int signum)
{
    cout<<endl<<"Fallo en Sensor"<<endl;
    cola_sensor.receive(pid_sensor, TAMCADENA*sizeof(char));
}
```



De esta manera obtenemos algo como lo mostrado en la siguiente imagen:

A terminal window titled 'diego@diego-VirtualBox: ~/Escritorio/Practica-2' with a dark purple background. It displays two lines of white text: 'Los limites del deposito son el mayor: 10000 y el menor: 1000' and 'Nivel del tanque, Estado de la valvula: 05400 ON' followed by a small white cursor block.

```
diego@diego-VirtualBox: ~/Escritorio/Practica-2  
  
Los limites del deposito son el mayor: 10000 y el menor: 1000  
Nivel del tanque, Estado de la valvula: 05400 ON █
```

Donde los valores del nivel del tanque y el estado de la válvula se actualizan.

## Código del programa

A continuación vamos a mostrar todos los códigos de los diferentes archivos.

A parte de todos los archivos que vamos a mostrar disponemos del archivo Makefile que nos servirá para compilar dichos archivos y los archivos aportados por el profesorado relacionados con las colas y con la memoria compartida, los cuales no van a ser mostrados en el informe.

### Control.cpp

```
1 //Recibe una cadena del proceso Sensor
2 //IMPORTANTE: Debe ejecutarse primero este proceso
3 //pues es este quien crea la cola de mensajes
4
5
6 #include <iostream>
7 #include "colamsq.hpp"
8 #include <string.h>
9 #include <stdlib.h>
10 #include <signal.h>
11 #include <unistd.h>
12 #define TAMCADENA 15
13 #define Nivel_MIN 1000
14 #define Nivel_MAX 10000
15 using namespace std;
16
17 char pid_sensor[TAMCADENA], pid_valvula[TAMCADENA], proceso_valvula[TAMCADENA], proceso_sensor[TAMCADENA];
18
19 colamsq cola_sensor("nivel_agua", CREAT, RDONLY, TAMCADENA*sizeof(char)); //Tienen qe ser declaradas comovariabls globales
20
21 colamsq cola_valv("llave", CREAT, WRONLY, TAMCADENA*sizeof(char)); //para que el manejador las pueda eliminar
22
23 colamsq cola_vive("proceso", CREAT, RDWR, TAMCADENA*sizeof(char));
24
25 void manejador(int signum); // declaración del manejador de señales
26 void alarma_sensor(int signum);
27 void alarma_valvula(void alarma_sensor(int signum))
28
29 int main (int argc, char *argv[])
30 {
31     int nivel, nivelMax, nivelMin;
32
33     if(argc==3)
34     {
35         if(atoi(argv[1])>atoi(argv[2]))
36         {
37             nivelMax=atoi(argv[1]);
38             nivelMin=atoi(argv[2]);
39         }
40         else if (atoi(argv[1])<atoi(argv[2]))
41         {
42             nivelMax=atoi(argv[2]);
43             nivelMin=atoi(argv[1]); //Introduce los límites del deposito independientemente del orden y solo si son lógicos
44         }
45         else
46         {
47             nivelMax=Nivel_MAX;
48             nivelMin=Nivel_MIN;
49         }
50     }
51
52     else
53     {
54         nivelMax=Nivel_MAX;
55         nivelMin=Nivel_MIN; //Toma valores por defecto
56     }
57 }
```





## Depósito.cpp

```
1 |
2 //Este programa representa el comportamiento del deposito
3 //lee el estado de la valvula y escribe el nivel del deposito
4
5 #define TAMCADENA 15
6 #include <stdlib.h>
7 #include <unistd.h>
8 #include <string.h>
9 #include <signal.h>
10 #include <iostream>
11 #include "semaforo.hpp"
12 #include "memocomp.hpp"
13 #define caudalEntrada 300
14 #define caudalSalida 100
15
16 using namespace std;
17
18 int pid_valvula, pid_sensor;
19
20 semaforo s_deposito("sem1", 0);
21 semaforo s_sensor("sem2", 1);
22
23 memocomp estado_llave("abre", CREAT, RDWR, TAMCADENA*sizeof(char));
24 memocomp estado_deposito("nivel", CREAT, RDWR, TAMCADENA*sizeof(char));
25
26 void manejador (int signum);
27
28 int main(int argc, char *argv[])
29 {
30     char *cadena_valvula, * p_aux; // puntero a la cadena en m
31     int nivel=0;
32     int Qentrada, Qsalida;
33
34     if(argc==3) // Asignamos el valor del c
35     { // y de salida correctamente
36         // argumentos introducidos
37         if(atoi(argv[1]) > atoi(argv[2]))
38         {
39             Qentrada=atoi(argv[1]);
40             Qsalida=atoi(argv[2]);
41         }
42         else if(atoi(argv[1]) < atoi(argv[2]))
43         {
44             Qentrada=atoi(argv[2]);
45             Qsalida=atoi(argv[1]);
46         }
47         else
48         {
49             Qentrada=caudalEntrada;
50             Qsalida=caudalSalida;
51         }
52     }
53     else
54     {
55         Qentrada=caudalEntrada;
56         Qsalida=caudalSalida;
57     }
```



*Sensor.cpp*

```

2 //Manda una cadena al proceso Control
3 // Debe ejecutarse antes Control.
4 //pues es este quien crea la cola de mensajes
5
6
7 #include <iostream>
8 #include <string.h>
9 #include <signal.h>
10 #include <unistd.h>
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include "colamsg.hpp"
14 #include "semaforo.hpp"
15 #include "memocomp.hpp"
16 #define TAMCADENA 15
17 using namespace std;
18
19 void manejador(int signum);
20
21 char altura_agua[TAMCADENA], *p_aux, miProceso[TAMCADENA];
22 colamsg sensor("nivel_agua", OPEN, WRONLY);
23
24 semaforo s_sensor("sem2");
25
26 memocomp estado_deposito("nivel", OPEN, RDWR, TAMCADENA*sizeof(char));
27
28
29 int main ()
30 {
31     p_aux = (char *)estado_deposito.getPointer();
32
33     sprintf(miProceso, "%d", getpid());
34     sensor.send(miProceso, TAMCADENA*sizeof(char)); //enviamos el pid de sensor
35
36
37     system("clear"); //Limpia la pantalla una vez se inicie el programa
38     cout<<"\n\n\n\n\n";
39
40     cout<<"Nivel del deposito:          ";
41
42     signal(SIGINT, manejador);
43
44     while(true)
45     {
46         s_sensor.down();
47
48         strcpy(altura_agua, p_aux);
49         sensor.send(altura_agua, TAMCADENA*sizeof(char));
50
51         printf("\b\b\b\b\b\b%05d", atoi(p_aux));
52         fflush(stdout); //actualizamos el nivel del deposito sin tener que imprimir una linea cada vez
53     }
54 }
55
56
57 void manejador(int signum) //cuando recibimos la señal ^C cierra los semaforos la memoria compartida y la cola de mensajes
58 {
59     cout<<endl<<"Fin de programa Sensor"<<endl;
60
61     sprintf(miProceso, "%d", -1);
62
63     sensor.send(miProceso, TAMCADENA*sizeof(char)); //envia -1 como señal de fin de proceso
64
65     sensor.close();
66
67     s_sensor.close();
68
69     estado_deposito.cerrar();
70     exit(0);
71 }
72
73

```

## Válvula.cpp

```
1
2 //Recibe una cadena al proceso Control
3 // Debe ejecutarse antes Control.
4 //pues es este quien crea la cola de mensajes
5
6 #include <iostream>
7 #include <string.h>
8 #include <signal.h>
9 #include <unistd.h>
10 #include <stdlib.h>
11 #include "colamsg.hpp"
12 #include "semaforo.hpp"
13 #include "memocomp.hpp"
14 #define TAMCADENA 15
15
16 using namespace std;
17
18     char miCadena[TAMCADENA], miProceso[TAMCADENA], *p_aux;
19
20     colamsg valvula("llave", OPEN, RDONLY);
21     colamsg cola_vive("proceso", OPEN, RDWR, TAMCADENA*sizeof(char));
22
23     semaforo s_deposito("sem1");
24
25     memocomp estado_llave("abre", OPEN, RDWR, TAMCADENA*sizeof(char));
26
27     void manejador (int signal); // declaración del manejador de señales
28
29 int main ()
30 {
31     sprintf(miProceso, "%d", getpid());
32     cola_vive.send(miProceso, TAMCADENA*sizeof(char)); //enviamos el pid de valvula
33
34     p_aux = (char *)estado_llave.getPointer();
35
36     sprintf(miProceso, "%d", 1);
37
38     signal (SIGINT, manejador);
39
40     system("clear"); //Limpia la pantalla una vez se inicie el programa
41     cout<<"\n\n\n\n\n\n";
42
43     cout<<"Estado valvula:\t ";
44
45     while(true)
46     {
47         valvula.receive(miCadena, TAMCADENA*sizeof(char));
48         strcpy(p_aux, miCadena);
49
50         printf("\b\b\b\b%s", p_aux);
51         fflush(stdout); //actualizamos el estado de la valvula sin tener que imprimir una linea cada vez
52
53         cola_vive.send(miProceso, TAMCADENA*sizeof(char)); //envia 1 todo en orden
54
55         s_deposito.up();
56     }
57
58
59 void manejador(int signal) //cuando recibimos la señal ^C cierra los semaforos la memoria compartida y la cola de mensajes
60 {
61     cout<<endl<<"Fin de programa Valvula"<<endl;
62
63     sprintf(miProceso, "%d", -1);
64     cola_vive.send(miProceso, TAMCADENA*sizeof(char));
65
66     valvula.close();
67     cola_vive.close();
68
69     s_deposito.close();
70
71     estado_llave.cerrar();
72
73     exit(0);
74 }
75
```

## Ejemplo de ejecución



The image shows four terminal windows arranged in a 2x2 grid, all with the title bar 'diego@diego-VirtualBox: ~/Escritorio/Practica-2'. The top-left window displays: 'Los linites del deposito son el mayor: 10000 y el menor: 1000' and 'Nivel del tanque, Estado de la valvula: 08000 OFF'. The top-right window displays: 'El caudal de entrada al deposito es 300 y el menor 100' and 'Nivel del tanque, Estado de la valvula: 08000 OFF'. The bottom-left window displays: 'Estado valvula: OFF'. The bottom-right window displays: 'Nivel del deposito: 08000'.

```
diego@diego-VirtualBox: ~/Escritorio/Practica-2
Los linites del deposito son el mayor: 10000 y el menor: 1000
Nivel del tanque, Estado de la valvula: 08000  OFF

diego@diego-VirtualBox: ~/Escritorio/Practica-2
El caudal de entrada al deposito es 300 y el menor 100
Nivel del tanque, Estado de la valvula: 08000  OFF

diego@diego-VirtualBox: ~/Escritorio/Practica-2
Estado valvula: OFF

diego@diego-VirtualBox: ~/Escritorio/Practica-2
Nivel del deposito: 08000
```

## **Bibliografía**

- Nuestra mayor fuente para la realización de este trabajo ha sido los materiales aportados en clase, tanto en la explicación de la práctica como los apuntes de teoría.
- [www.cplusplus.com](http://www.cplusplus.com)