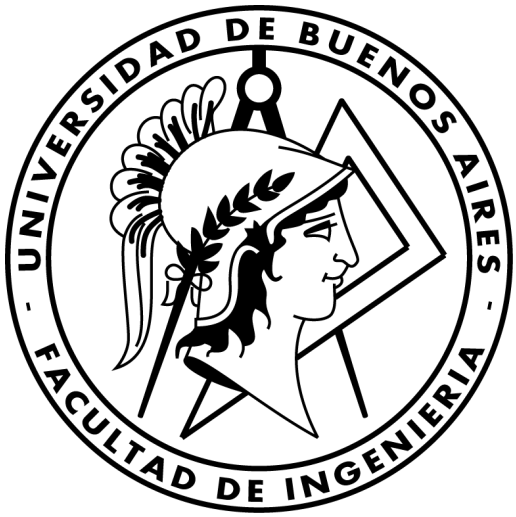
Segundo Proyecto – Técnicas de Programación Concurrente (75.59)

-Cliente-Servidor-



Facultad de Ingeniería

1er Cuatrimestre - 2013

Alumnos:

- Magnaghi, Pablo Marcelo - Padrón: 88126
- Montiel, Gonzalo - Padrón: 89932
- Schmid, Pablo Hernan – Padrón : 88760

Índice de contenido

Análisis del Problema 3

Compilación y ejecución 3

- Modo de uso 3

Explicación general de la implementación 5

Resolución de Tareas 6

- División del programa en procesos 6
- Comunicación entre procesos 6
- División del programa en procesos 6

Diagrama de Clases de la Solución 8

Análisis del problema

Se deben realizar dos aplicaciones. Una correspondiente a un Servidor y la otra correspondiente a un Cliente. El servidor deberá soportar la conexión de 1 o más clientes logrando así una aproximación más real de aplicaciones cliente-servidor.

El servidor será el encargado de administrar una base de datos de personas.

Se deberá modelar el servidor y clientes, estableciendo los mecanismos de comunicación y concurrencia que se adecuen a cada caso, permitiendo a los clientes hacer altas, bajas, modificaciones y consultas sobre la base de datos que administrará el servidor.

Compilación y ejecución

Para la correcta compilación del código fuente, debemos acceder a una consola y situarnos en la carpeta root del proyecto. Aquí escribimos:

```
$ sudo ./compilar.sh
```

Esto generará dos ejecutables. uno en la carpeta Servidor y otro en la carpeta Cliente. Para ejecutar el servidor y el cliente vamos a sus respectivas carpetas y tipeamos:

```
$ ./MainServer
```

```
$ ./MainCliente
```

Modo de uso y sintaxis de los comandos

Los registros están formados por nombre, dirección y teléfono. El nombre es utilizado como clave para identificar un usuario, por ende debe ser único. Los ejemplos se deben interpretar como interacciones consecutivas.

1. Insertar'

1.1 Sintaxis

```
> insertar -n <nombre> -d <direccion> -t <telefono>
```

```
> I -n <nombre> -d <direccion> -t <telefono>
```

Parámetros obligatorios: **-n**.

El orden es indistinto.

1.2 Ejemplos

```
> insertar -n gonzalo -d calle falsa 1234
```

Insertará el registro de nombre 'gonzalo' con dirección 'calle falsa 1234' y teléfono ''.

```
> insertar -n gonzalo
```

Dará error por duplicado

```
> insertar -t 123456 -d otra calle 1234
```

Dará error por parámetros oligatorios faltantes (nombre).

```
> insertar -n juan -d otra calle 1234 -t 345345
```

Insertará el registro de nombre 'juan' con dirección 'otra calle 1234' y teléfono '345345'.

2. Modificar

2.1 Sintaxis

```
> modificar -m <nombre_a_modif> -n <nuevo_nombre> -t <nuevo_telefono> -d <nueva_direccion>
> M -m <nombre_a_modif> -n <nuevo_nombre> -t <nuevo_telefono> -d <nueva_direccion>
```

Parámetros obligatorios: **-m.**

El orden es indistinto.

2.2 Ejemplos:

```
> modificar -m gonzalo -n juan
```

Dará error pues 'juan' ya existe.

```
> modificar -m gonzalo -n pablo -t 456456
```

Modificará el usuario 'gonzalo' al nombre 'pablo' y le pondrá el teléfono '456456' donde antes no había nada.

```
> modificar -n monalisa -d lalala
```

Dará error pues no se especificó a quien modificar (-m).

3. Consultar

3.1 Sintaxis

```
>  consultar [-o OR/AND] [-n <nombre>] [-t <telefono>] [-d <direccion>]
```

Parámetros obligatorios: ninguno.

Sin parámetros devuelve todos los registros.

El parámetro -o sirve para hacer un 'and' o un 'or' lógico entre las condiciones ingresadas. Por defecto es OR.

El orden es indistinto.

3.2 Ejemplos

```
> consultar -n juan
```

Devolverá los datos de todos los usuarios que contengan 'juan' en su nombre.

```
> consultar -n juan -t 456456
```

Devolverá los datos de todos los usuarios cuyo nombre contenga 'juan' o su teléfono contenga '456456'. En este caso serán 'juan' y 'pablo'.

```
> consultar -n juan -t 456456 -o AND
```

Devolverá los datos de todos los usuarios cuyo nombre contenga 'juan' Y su teléfono contenga '456456'. En este caso será vacío.

Nota: el operador booelando se aplica para todos los campos que se ingresen, no están soportadas consultas booleanas complejas.

```
> consultar
```

Devolverá los datos de todos los usuarios del sistema.

4. Salir

```
>  salir
>  S
```

5. Ayuda

```
>  help
>  H
```

Explicación de la solución

Servidor

El Servidor administrará la base de datos de personas. Al correr el server, automaticamente se creará una cola de mensajes que será la encargada de recibir todos los mensajes de todos los clientes. Todos los mensajes que lleguen allí serán leídos por el servidor y de acuerdo al tipo de mensaje que sea el servidor procederá a procesarlos. El servidor se bloqueará leyendo la cola de mensajes hasta obtener el primer mensaje. Una vez recibido, lo procesará y lo responderá a quien corresponda. Luego se bloqueará leyendo el siguiente mensaje. Este loop continuará hasta que le llegue al server una señal de salida (SGINT). En este caso se enviará a todos los clientes conectados un mensaje a cada uno de que el servidor se cayó. Así de esta manera los clientes sabrán que el servidor ya no está disponible y deberán interrumpir sus aplicaciones.

Cliente

Una vez que se corre la aplicación, automáticamente el cliente verificará si existe la cola del servidor. Si no existe, sale. En caso exitoso, creará la misma cola de mensajes que el servidor y aparte una cola propia identificada con su Pid, para recibir respuestas del servidor. A continuación le enviará un mensaje de "acá estoy" al servidor mediante la cola del servidor. Ese mensaje de bienvenida es importante para que el servidor no solamente detecte que tiene un nuevo cliente conectado, sino que sepa cual es el Pid del cliente. El Pid del cliente se usará para que el servidor pueda mandar mensajes en la cola de dicho cliente.

Una vez establecida esta conexión, el cliente se quedará esperando a que se ingrese por entrada estándar la siguiente instrucción a ejecutar. Entonces, mientras el mensaje ingresado por línea de comandos no sea el de "salir", el cliente enviará un mensaje al servidor y se

quedará esperando la respuesta por la cola identificada con su pid.

Resolución de Tareas

1) División del programa en procesos

Los procesos involucrados en el proyecto son :

- **Proceso servidor**
- **Procesos cliente**
  - **Proceso principal cliente**
  - **Proceso que chequea cierre del servidor**

2) Comunicación entre procesos

Escritura de cliente a servidor	Cola de mensajes única
Escritura de servidor a cliente	Cola de mensajes por cliente
Servidor se cierra y avisa a clientes	Señales y cola de mensajes (mensaje de EXIT)
Usuario cierra servidor	Señales

3) Mecanismos de Concurrencia

Para cada uno de las comunicaciones que hay entre los procesos, analizamos una serie problemas de concurrencia que se plantean a continuación, junto con su solución.

Cliente se conecta por primera vez

Cuando el cliente se conecta por primera vez, necesita saber si el servidor está corriendo o no. Para esto, prueba abrir la cola con la opción IPC_EXCL. Si la cola existe, devuelve un error. Esto es aprovechado para dicho chequeo y se puede saber cuando el cliente tiene donde escribir y cuando no.

Notificar al servidor de un nuevo cliente

Una vez asegurados de que existe la cola, el cliente envía un mensaje especial al servidor para avisarle de su existencia. El servidor entonces crea una nueva cola de mensajes para comunicación servidor-cliente con dicho cliente nuevo. En las próximas peticiones de este cliente el servidor utilizará la cola que acaba de crear.

Envío de comandos del cliente al servidor

Esto se realiza por medio de la cola de mensajes única del servidor, en la cual se van

agregando todas las peticiones para que el servidor las procese de a una.

Envío de respuestas del servidor al cliente

Luego de procesar una petición, el servidor responde al cliente por la cola particular de cada cliente. El cliente recibe esta respuesta y la procesa también, mostrando lo que deba mostrar.

Cierre del servidor

Aquí se utilizan señales. La idea es poder terminar el servidor mediante la señal SIGINT, sobreescribiendo el handler y terminando de manera elegante, sin dejar nada incoherente.

Así pues, si desde una terminal se tipea Ctrl+C, el servidor terminará correctamente y avisará a sus clientes.

Aviso de cierre del servidor a los clientes

Cuando el servidor se cierra, utiliza dos cosas para avisar a sus clientes. En primer lugar, escribe en la cola de cada cliente un mensaje de EXIT, el cual los clientes procesan en un proceso aparte del principal, y cuando se recibe este mensaje se imprime al usuario una advertencia de que el servidor se cerró.

A su vez, se le envía una señal a cada cliente para que dejen de estar en donde estaban (posiblemente esperando I/O) y modifiquen un flag de 'servidor cerrado'.

Identificar cierre del servidor en el cliente

Al iniciar el cliente se creará un proceso hijo, el cual se quedará esperando, en la cola de mensajes recibidos del servidor, un mensaje del tipo "EXIT". Este mensaje indicará al cliente que el server se cerró y que el cliente debe finalizar. Al recibir este mensaje el proceso hijo se desbloqueará y le pedirá al usuario que ingrese "cualquier tecla" para finalizar la aplicación.

Diagrama de Clases de la Solución