Práctica 5 - Sistema Cliente-Servidor Distribuido

Descripción

Implementación de un sistema distribuido para procesamiento de tareas usando sockets síncronos. El sistema consta de:

- **Servidor de Tareas**: Gestiona el multibuffer y distribuye tareas a los controladores
- **Servidor de Matriz**: Gestiona el acceso concurrente a la matriz de resultados
- **Controladores**: Procesos cliente que obtienen tareas del servidor de tareas y registran resultados en el servidor de matriz

⚠️ IMPORTANTE: Ejercicios Previos

Antes de realizar la práctica 5, **debes completar los ejercicios previos** del enunciado:

- **Ejercicio 1**: Cliente y Servidor básico (un solo cliente)
- **Ejercicio 2**: Servidor Multicliente (múltiples clientes concurrentes)

Estos ejercicios te ayudarán a entender la comunicación mediante sockets antes de abordar el sistema completo.

Estructura de Directorios

```
practica5/
├── # Práctica 5 - Sistema distribuido principal
├── ServidorTareas.cpp
├── ServidorMatriz.cpp
├── Controlador.cpp
│
├── # Ejercicios previos
├── Cliente.cpp            (Ejercicio 1)
├── Servidor.cpp           (Ejercicio 1)
├── ServidorMulticliente.cpp (Ejercicio 2)
│
├── # Archivos comunes
├── tarea.hpp
├── Makefile
├── lanzar_sistema.sh
├── tareas.txt
│
├── Socket/
│   ├── Socket.hpp
│   └── Socket.cpp
│
└── librerias/
```

```
├── Monitores/
│   ├── BufferTareas.hpp
│   ├── BufferTareas.cpp
│   ├── GestionResultados.hpp
│   └── GestionResultados.cpp
└── MultiBuffer/
    ├── MultiBuffer.hpp
    └── MultiBuffer.cpp
```

Compilación

Compilar la práctica 5 completa:

```bash
make
```

Compilar solo los ejercicios previos:

```bash
make ejercicios
```

Compilar todo:

```bash
make all
make ejercicios
```

Para compilar en hendrix, descomentar la línea del Makefile:

```makefile
SOCKETSFLAGS=-lsocket -lnsl
```

Ejercicios Previos (OBLIGATORIOS)

Ejercicio 1: Cliente-Servidor Básico

Objetivo: Aprender comunicación básica mediante sockets con un solo cliente.

Terminal 1 - Servidor:

```bash
./Servidor 3000
```

Terminal 2 - Cliente (mismo ordenador):

```bash
./Cliente localhost 3000
```

Terminal 2 - Cliente (ordenador remoto):

```bash
# Obtener IP del servidor
host hendrix01.cps.unizar.es  # Ejemplo: 155.210.154.205

# Conectar desde otro ordenador
./Cliente 155.210.154.205 3000
```

Funcionamiento:

- Escribe frases en el cliente
- El servidor cuenta las vocales y responde
- Escribe "END OF SERVICE" para terminar

Ejercicio 2: Servidor Multicliente

Objetivo: Aprender a atender múltiples clientes concurrentemente con threads.

Terminal 1 - Servidor:

```bash
./ServidorMulticliente 3000
```

Terminales 2, 3, 4... - Múltiples Clientes:

```bash
# Cliente 1
./Cliente localhost 3000

# Cliente 2 (en otro terminal)
./Cliente localhost 3000

# Cliente 3 (en otro terminal u ordenador)
./Cliente 155.210.154.205 3000
```

Funcionamiento:

- Cada cliente es atendido por un thread separado
- Todos pueden enviar mensajes simultáneamente
- El servidor se cierra automáticamente después de 60 segundos sin actividad

Ejecución en Local (Modo Prueba)

Práctica 5 - Sistema Distribuido Completo

Una vez completados los ejercicios previos, puedes ejecutar la práctica 5:

Opción 1: Script automático

```bash
chmod +x lanzar_sistema.sh
./lanzar_sistema.sh
```

Opción 2: Manual

Terminal 1 - Servidor de Tareas:

```bash
./ServidorTareas 3000
```

Terminal 2 - Servidor de Matriz:

```bash
./ServidorMatriz 3001
```

Terminales 3-12 - Controladores (10 en total):

```bash
./Controlador localhost 3000 localhost 3001
```

Repetir para cada uno de los 10 controladores.

Ejecución Distribuida

En el servidor de tareas (máquina A):

```bash
./ServidorTareas 3000
```

En el servidor de matriz (máquina B):

```bash
./ServidorMatriz 3001
```

En cada máquina controladora (máquinas C1, C2, ..., C10):

```bash
./Controlador <IP_maquina_A> 3000 <IP_maquina_B> 3001
```

Ejemplo:

```bash
./Controlador 155.210.154.205 3000 155.210.154.206 3001
```

Protocolo de Comunicación

Servidor de Tareas ↔ Controlador

1. **Controlador** se conecta al servidor
2. **Servidor** envía tarea: `"tipo,carga"` (ej: `"t1,45.3"`)
3. Cuando termina: **Servidor** envía `"TF,0"`
4. **Controlador** responde con `"END"`

Servidor de Matriz ↔ Controlador

1. **Controlador** se conecta al servidor
2. Por cada tarea procesada, **Controlador** envía: `"tipo,exito,carga"` (ej: `"t1,1,45.3"`)
   - `exito = 1` si la tarea tuvo éxito, `0` si falló
3. Cuando termina: **Controlador** envía `"TF"`
4. **Servidor** responde con `"END"` cuando todos los controladores han terminado

Formato del archivo tareas.txt

```
t1,45.5
t2,30.2
t3,60.8
t1,25.4
...
```

Cada línea contiene: `tipo_tarea,carga_de_trabajo`

Salida Esperada

El **Servidor de Matriz** mostrará al final:

```
==========================================
  RESULTADOS DEL PROCESAMIENTO
==========================================
Tarea |  Total | Exito  | Tiempo
------+--------+--------+--------
t1    |    150 |    142 | 6785.34
t2    |    200 |    186 | 9234.12
t3    |    100 |     90 | 5678.90
==========================================
Tasas de exito:
  t1: 94.7%
  t2: 93.0%
  t3: 90.0%
==========================================
```

Archivos de Log (al usar script)

- `servidor_tareas.log`: Log del servidor de tareas
- `servidor_matriz.log`: Log del servidor de matriz (incluye resultados finales)
- `controlador_0.log` ... `controlador_9.log`: Logs de cada controlador

Monitorización en Tiempo Real

```bash
# Ver progreso del servidor de tareas
tail -f servidor_tareas.log

# Ver progreso del servidor de matriz
tail -f servidor_matriz.log

# Ver progreso de un controlador específico
tail -f controlador_0.log
```

Limpieza

```bash
make clean
rm -f *.log
```

Notas Importantes

1. Los **10 controladores deben conectarse** antes de que el sistema comience a procesar tareas
2. El orden de inicio recomendado es:
   - Primero: Servidor de Tareas y Servidor de Matriz
   - Después: Todos los Controladores
3. Para ejecución distribuida, verificar que los puertos estén abiertos en los firewalls
4. Usar `ifconfig` o `ip addr` para obtener la IP de cada máquina

Obtener IP de una máquina

```bash
# En Linux
ip addr
# o
ifconfig

# Para hendrix
host hendrix01.cps.unizar.es
```

Solución de Problemas

Error: "Error en Bind"

- El puerto ya está en uso. Cambia el puerto o espera a que se libere

Error: "No se pudo conectar al servidor"

- Verifica que el servidor esté ejecutándose
- Verifica la IP y puerto correctos
- Verifica que no haya firewall bloqueando la conexión

Los controladores no empiezan a procesar

- Asegúrate de que los 10 controladores estén conectados a ambos servidores

Autores

Práctica 5 - PSCD
Universidad de Zaragoza