



ACTIVIDAD 01.

SOCKETS

DESARROLLO DE SOFTWARE EN
SISTEMAS DISTRIBUIDOS

Equipo Docente:
Ing. Diego Andrés Azcurra
Lic. Marcos Amaro

2024

Estudiante:

Robles Flores Sergio Simon
DNI: 35355978

Actividad 01. Sockets

Datos del Estudiante

- **Nombre:** Sergio Simón Robles Flores
- **DNI:** 35355978

Link al repositorio:

<https://github.com/ImNotThrasher/8627-distribuidos>

Cómo Clonar el Repositorio

Para clonar este repositorio, utilizar el siguiente comando en la terminal o consola:

```
git clone https://github.com/ImNotThrasher/8627-distribuidos.git
```

Requerimientos

1. **Desarrollar un servidor en lenguaje C utilizando sockets, implementando las siguientes funcionalidades:**
 - **Generador de nombres de usuario:** El servidor debe generar un nombre de usuario indicando la longitud deseada (debe validar que no sea menor a 5 ni mayor a 15). La cadena debe alternar entre vocales y consonantes. El servidor elegirá al azar si empieza por una vocal o consonante.
 - **Generador de contraseñas:** El servidor debe generar una contraseña alfanumérica, incluyendo mayúsculas y minúsculas, indicando la longitud deseada (debe validar que sea mayor o igual a 8 y menor a 50).
2. **Desarrollar un cliente en lenguaje C:** El cliente debe conectarse al servidor y, mediante un menú, podrá elegir entre generar un nombre de usuario o una contraseña, enviando la longitud deseada. Debe mostrar la respuesta generada por el servidor o un mensaje de error en caso de validación.
3. **Desarrollar un segundo cliente en un lenguaje diferente:** El cliente debe tener la misma funcionalidad que el primero pero estar implementado en un lenguaje diferente, como Kotlin, Go, JavaScript (Node.js), Python, etc.

Introducción

La aplicación se enfoca principalmente en la generación de nombres de usuario y contraseñas. Además de la implementación en C, se optó por desarrollar un cliente adicional en Python debido a su facilidad para scripting, su rapidez para el desarrollo y el hecho de que no necesita compilación, ofreciendo así un contraste con el enfoque en C.

Para el desarrollo se utilizaron las siguientes herramientas:

- Visual Studio Code como entorno de desarrollo integrado (IDE).
- Git para el control de versiones.
- GCC como compilador para C, instalado a través de MinGW.
- PyInstaller para convertir el script de Python en un ejecutable .exe.

Desarrollo Modular

Para mejorar la legibilidad del código y facilitar su reutilización, se decidió desarrollar el programa en varios archivos separados y en distintas funciones. Esto permitió escalar el proyecto desde la funcionalidad inicial.

Parte desarrollada en C:

- **socket_utils.c:** Contienen funciones auxiliares para la gestión de sockets.
- **cliente.c:** Implementa el cliente.
- **servidor.c:** Implementa el servidor.
- **config.c:** Incluye las variables de configuración como IP, puerto, longitud de nombre de usuario y contraseña, y opciones de reconexión.

Ubicación de Archivos

El servidor y el cliente en C están ubicados en la carpeta [/Servidor y Cliente en C](#).

El cliente adicional en Python se encuentra en la carpeta [/Cliente en Python](#).

Para facilitar la prueba de la aplicación, se proporcionan archivos ejecutables .exe en la carpeta [/Ejecutables](#). Se puede ejecutar tanto el servidor como ambos clientes (en C y Python) utilizando el script ejecutar_todo.bat en [/Ejecutables/ejecutar_todo.bat](#), o ejecutarlos individualmente.

Compilación Manual

Si se prefiere compilar los archivos manualmente, se puede hacer siguiendo las siguientes indicaciones:

Servidor en C: Dentro de la carpeta [/Servidor y Cliente en C](#) se debe ejecutar el siguiente comando en la terminal:

```
gcc servidor.c socket_utils.c -o servidor -lws2_32
```

Cliente en C: Desde la misma carpeta, ejecutar:

```
gcc cliente.c socket_utils.c -o cliente -lws2_32
```

Cliente en Python: Desde la carpeta [/Cliente en Python](#) se debe ejecutar el siguiente comando para correr el script directamente:

```
python cliente.py
```

Si se prefiere generar un ejecutable, se puede usar PyInstaller:

```
pyinstaller --onefile cliente.py
```

Desarrollo Inicial: Conexión Básica con Sockets

El punto de partida fue establecer una simple conexión utilizando sockets, donde el servidor esperaba mensajes enviados por el cliente. En esta fase inicial, el servidor recibía cualquier mensaje del cliente y respondía de manera genérica, sin implementar lógica adicional.

Tipos de Solicitudes en la Comunicación

Una vez establecida la comunicación básica, el siguiente paso fue mejorar la capacidad del servidor para controlar el tipo de mensajes recibidos. Se introdujeron parámetros específicos en la comunicación, como USER para identificar al usuario, PASS para la contraseña, seguido de la longitud de las cadenas de texto (Entonces el cliente le envía dos parámetros USER o PASS y la longitud, ej USER 5). Estos parámetros permiten al servidor diferenciar entre distintos las distintas solicitudes y responder de la manera correspondiente. De esta forma, el servidor puede generar un nombre de usuario o generar una contraseña, dependiendo la longitud solicitada por el cliente. En el caso de que la longitud este fuera de rango, el cliente recibirá un error y podrá realizar una nueva solicitud al servidor.

Escucha Permanente del Servidor

Una de las mejoras más importantes del servidor fue la implementación de escucha permanente, si bien no está especificado en la consigna de la actividad, esta funcionalidad permite al servidor manejar múltiples clientes de manera concurrente.

Manejo de Conexiones Múltiples

La siguiente mejora fue la introducción de la capacidad del servidor para manejar múltiples conexiones simultáneamente. Se implementó un sistema de manejo de hilos (threads) que permite que el servidor atienda varios clientes al mismo tiempo. Cada cliente que se conecta al servidor recibe su propio hilo dedicado.

Reconexión Automática del Cliente

Lo último que se implementó fue la reconexión automática del cliente en caso de que se perdiera la conexión con el servidor esperando una respuesta. En caso de una interrupción temporal de la red o de una caída del servidor, el cliente intenta reconectarse automáticamente.

Flujo de Ejecución

Flujo de Ejecución: Servidor

1. **Inicio del Servidor:** El servidor se inicializa y configura el socket para escuchar conexiones en un puerto específico.
2. **Modo de Espera:** El servidor entra en un modo de espera, escuchando conexiones entrantes de clientes.
3. **Aceptación de Conexiones:** Cuando un cliente se conecta, el servidor acepta la conexión y asigna un hilo dedicado para manejar la comunicación con ese cliente.
4. **Recepción y Procesamiento de Mensajes:** El servidor recibe mensajes del cliente, analiza los parámetros (user, pass, longitud) y realiza las acciones correspondientes, como validar la longitud y generar usuarios o contraseñas.
5. **Respuesta al Cliente:** Dependiendo del mensaje recibido y los parámetros procesados, el servidor envía una respuesta apropiada al cliente.
6. **Escucha Permanente:** El servidor continúa en escucha permanente, aceptando nuevas conexiones mientras sigue interactuando con clientes ya conectados.
7. **Manejo de Desconexiones:** Si un cliente se desconecta, el servidor cierra el hilo correspondiente y continúa en espera de nuevas conexiones.

Flujo de Ejecución: Cliente

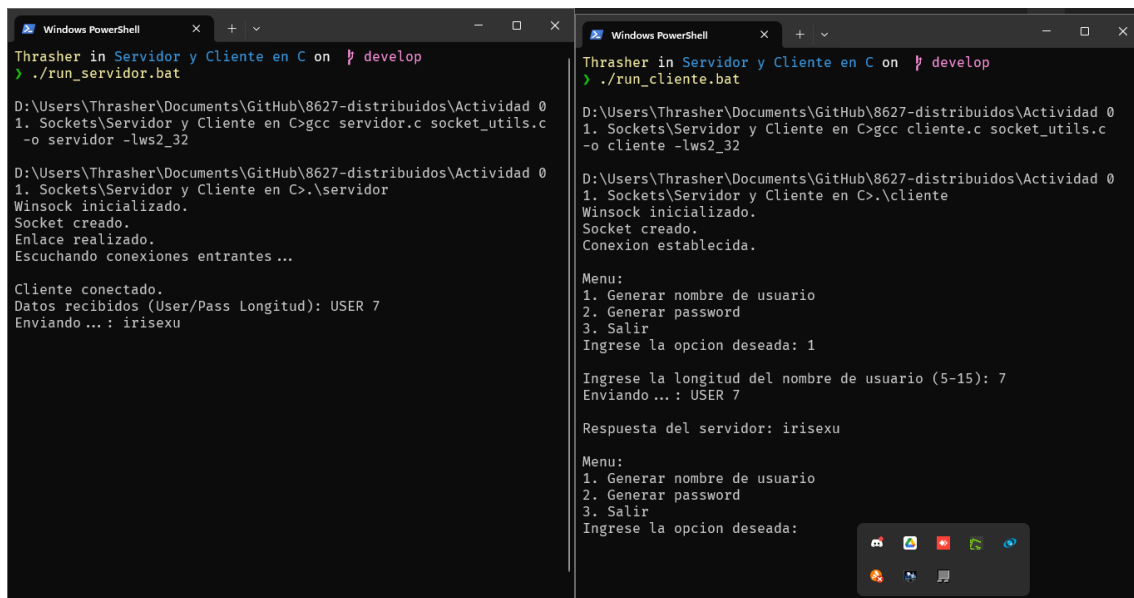
1. **Inicio del Cliente:** El cliente se inicializa y configura para conectarse al servidor en un puerto y dirección IP específicos.
2. **Conexión al Servidor:** El cliente establece una conexión con el servidor utilizando sockets.
3. **Selección de Operación:** El cliente presenta un menú interactivo al usuario para elegir entre generar un nombre de usuario o una contraseña.
4. **Envío de Solicitud:** El cliente envía un mensaje al servidor con los parámetros necesarios (user, pass, longitud) según la operación seleccionada.
5. **Recepción de Respuesta:** El cliente recibe la respuesta del servidor, que puede ser un nombre de usuario generado, una contraseña, o un mensaje de error.
6. **Reconexión Automática:** Si la conexión con el servidor se interrumpe, el cliente intenta reconectarse automáticamente para restablecer la comunicación.
7. **Cierre de Conexión:** Una vez completada la operación, el cliente puede optar por desconectarse o realizar una nueva solicitud al servidor.

Casos de Prueba

A continuación, se detallan los distintos casos de prueba utilizados para validar la actividad. Cada caso de prueba se acompaña de una imagen que muestra los resultados esperados.

Caso de Prueba 1: Conexión Básica entre Cliente y Servidor

Se realiza una prueba de conectividad entre un cliente y un servidor.



```
Windows PowerShell
Thrasher in Servidor y Cliente en C on  develop
> ./run_servidor.bat

D:\Users\Thrasher\Documents\GitHub\8627-distribuidos\Actividad 0
1. Sockets\Servidor y Cliente en C>gcc servidor.c socket_utils.c
-o servidor -lws2_32

D:\Users\Thrasher\Documents\GitHub\8627-distribuidos\Actividad 0
1. Sockets\Servidor y Cliente en C>.\servidor
Winsock inicializado.
Socket creado.
Enlace realizado.
Escuchando conexiones entrantes ...

Cliente conectado.
Datos recibidos (User/Pass Longitud): USER 7
Enviando ... : irisexu

Windows PowerShell
Thrasher in Servidor y Cliente en C on  develop
> ./run_cliente.bat

D:\Users\Thrasher\Documents\GitHub\8627-distribuidos\Actividad 0
1. Sockets\Servidor y Cliente en C>gcc cliente.c socket_utils.c
-o cliente -lws2_32

D:\Users\Thrasher\Documents\GitHub\8627-distribuidos\Actividad 0
1. Sockets\Servidor y Cliente en C>.\cliente
Winsock inicializado.
Socket creado.
Conexion establecida.

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada: 1

Ingrese la longitud del nombre de usuario (5-15): 7
Enviando ... : USER 7

Respuesta del servidor: irisexu

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada:
```

Caso de Prueba 2: Pedido de Nombre de Usuario y Validación de Longitud

Se realiza varias peticiones de nombre de usuario y se valida longitud desde el servidor cuando la longitud esta fuera del rango.

```
Servidor en C
Datos recibidos (User/Pass Longitud): USER 5
Enviando...: odiko

Datos recibidos (User/Pass Longitud): USER 15
Enviando...: cenaseborocaquil

Datos recibidos (User/Pass Longitud): USER 4
Error en generarUsername: Error: La longitud del nombre de usuario debe estar entre 5 y 15.
Enviando...: Error: La longitud del nombre de usuario debe estar entre 5 y 15.

Datos recibidos (User/Pass Longitud): USER 16
Error en generarUsername: Error: La longitud del nombre de usuario debe estar entre 5 y 15.
Enviando...: Error: La longitud del nombre de usuario debe estar entre 5 y 15.

Datos recibidos (User/Pass Longitud): USER 5
Enviando...: jelid

Datos recibidos (User/Pass Longitud): USER 5
Enviando...: sehis

Datos recibidos (User/Pass Longitud): USER 5
Enviando...: dajav

|

Cliente en C
Ingrese la opcion deseada: 1

Ingrese la longitud del nombre de usuario (5-15): 5
Enviando...: USER 5

Respuesta del servidor: jelid

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada: 1

Ingrese la longitud del nombre de usuario (5-15): 5
Enviando...: USER 5

Respuesta del servidor: sehis

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada: 1

Ingrese la longitud del nombre de usuario (5-15): 5
Enviando...: USER 5

Respuesta del servidor: dajav

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada:
```

Caso de Prueba 3: Pedido de Contraseña y Validación de Longitud

Se realiza varias peticiones de contraseña y se valida longitud desde el servidor cuando la longitud esta fuera del rango.

```
Servidor en C
Datos recibidos (User/Pass Longitud): PASS 8
Enviando...: kL0whU9w

Datos recibidos (User/Pass Longitud): PASS 50
Enviando...: uLnkRoKSL7TMoceVC2ZuPMYgP2uq8jQM0cMSgTxwvNbsULJB

Datos recibidos (User/Pass Longitud): PASS 7
Error en generarPassword: Error: La longitud de la password debe estar entre 8 y 50.
Enviando...: Error: La longitud de la password debe estar entre 8 y 50.

Datos recibidos (User/Pass Longitud): PASS 51
Error en generarPassword: Error: La longitud de la password debe estar entre 8 y 50.
Enviando...: Error: La longitud de la password debe estar entre 8 y 50.

Datos recibidos (User/Pass Longitud): PASS 8
Enviando...: 1T7TXdff

Datos recibidos (User/Pass Longitud): PASS 8
Enviando...: 7BRBaTEO

Datos recibidos (User/Pass Longitud): PASS 50
Enviando...: nnY8t38g0ULRxxZkdLuAst9Q8pPUVIWc25jzW2Ls8IW21UFpmY

Datos recibidos (User/Pass Longitud): PASS 8
Enviando...: Ajs3TnWp

|

Cliente en C
Respuesta del servidor: 7BRBaTEO

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada: 2

Ingrese la longitud de la password (8-50): 50
Enviando...: PASS 50

Respuesta del servidor: nnY8t38g0ULRxxZkdLuAst9Q8pPUVIWc25jzW2Ls8IW21UFpmY

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada: 2

Ingrese la longitud de la password (8-50): 8
Enviando...: PASS 8

Respuesta del servidor: Ajs3TnWp

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada:
```

Caso de Prueba 4: Reinicio del Cliente y Reconexión

Se reinicia el cliente y se realiza una reconexión de forma automática.

```
Windows PowerShell
1. Sockets\Servidor y Cliente en C>.\servidor
Winsock inicializado.
Socket creado.
Enlace realizado.
Escuchando conexiones entrantes ...

Cliente conectado.
Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : bobuv

Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : zapaw

Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : uqaku

Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : rexew

Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : oqujo

Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : ehunu

Error al recibir datos: 10054
Cerrando conexion con el cliente ...

Cliente conectado.
Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : udopi

Windows PowerShell
3. Salir
Ingrese la opcion deseada:
^C;Desea terminar el trabajo por lotes (S/N)? s
Thrasher in Servidor y Cliente en C on  develop
> ./run_cliente.bat

D:\Users\Thrasher\Documents\GitHub\8627-distribuidos\Actividad 0
1. Sockets\Servidor y Cliente en C>gcc cliente.c socket_utils.c
-o cliente -lws2_32

D:\Users\Thrasher\Documents\GitHub\8627-distribuidos\Actividad 0
1. Sockets\Servidor y Cliente en C>.\cliente
Winsock inicializado.
Socket creado.
Conexion establecida.

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada: 1

Ingrese la longitud del nombre de usuario (5-15): 5
Enviando ... : USER 5

Respuesta del servidor: udopi

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada:
```

Caso de Prueba 5: Reinicio del Servidor y Reconexión

Se reinicia el servidor y se realiza una reconexión de forma automática.

```
Windows PowerShell
Enlace realizado.
Escuchando conexiones entrantes ...

Cliente conectado.
Datos recibidos (User/Pass Longitud): USER 7
Enviando ... : irisexu

Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : enuci

Datos recibidos (User/Pass Longitud): PASS 8
Enviando ... : Fb0rdA00

^C;Desea terminar el trabajo por lotes (S/N)? s
Thrasher in Servidor y Cliente en C on  develop
> ./run_servidor.bat

D:\Users\Thrasher\Documents\GitHub\8627-distribuidos\Actividad 0
1. Sockets\Servidor y Cliente en C>gcc servidor.c socket_utils.c
-o servidor -lws2_32

D:\Users\Thrasher\Documents\GitHub\8627-distribuidos\Actividad 0
1. Sockets\Servidor y Cliente en C>.\servidor
Winsock inicializado.
Socket creado.
Enlace realizado.
Escuchando conexiones entrantes ...

Cliente conectado.
Datos recibidos (User/Pass Longitud): USER 5
Enviando ... : bobuv

Windows PowerShell
Ingrese la opcion deseada: 1

Ingrese la longitud del nombre de usuario (5-15): 5
Error al enviar datos.Codigo de Error: 10054
Se perdio la conexion con el servidor. Intentando reconectar ...
Socket creado.
Fallo la conexion con el servidor. Codigo de Error: 10061
Reintento de conexion en 5 segundos ...
Socket creado.
Fallo la conexion con el servidor. Codigo de Error: 10061
Reintento de conexion en 5 segundos ...
Socket creado.
Conexion establecida.

Reconexion exitosa.

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada: 1

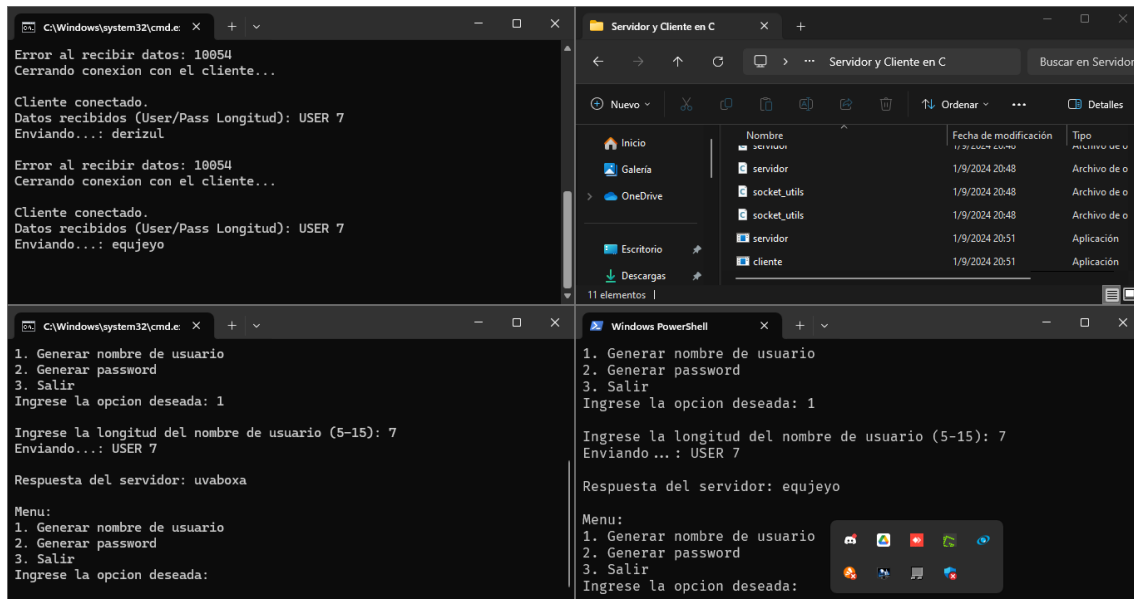
Ingrese la longitud del nombre de usuario (5-15): 5
Enviando ... : USER 5

Respuesta del servidor: bobuv

Menu:
1. Generar nombre de usuario
2. Generar password
3. Salir
Ingrese la opcion deseada:
```

Caso de Prueba 6: Múltiples Clientes Conectados al Servidor

Se conectan al servidor múltiples clientes programados en C.



Caso de Prueba 7: Múltiples Clientes en Distintos Lenguajes Conectados al Servidor

Se conectan al servidor múltiples clientes programados en C y Python.

