Programación Sobre Redes

Teo 13: Introducción a sockets(UNIX)

Nicolás Mastropasqua

October 22, 2020

Instituto Industrial Luis A. Huergo

Contenidos

1. Comunicación entre procesos: Repaso

2. Sockets

Comunicación entre procesos: Repaso

Pasaje de mensajes

Dos procesos se comunican a través del pasaje de mensaje entre ellos.

- Un proceso *P* envía un mensaje a otro proceso *Q*(**send**)
- El proceso *Q* recibe dicho mensaje (**receive**)

Mensajes: Bloqueante vs No Bloqueante

Bloqueante:

- Un proceso se bloquea al enviar un mensaje, se desbloquea cuando ese mensaje es recibido.
- Un proceso se bloquea al intentar recibir un mensaje, se mantiene bloqueado mientras el buzón esté vacío.

No Bloqueante:

- Un proceso envía el mensaje y sigue ejecutando, sin importar si este es recibido o no.
- Un proceso intenta recibir un mensaje y lo obtiene, si es que hay alguno esperando en el buzón, o obtiene un mensaje nulo en caso contrario

Sockets

Protocolos, APIs e Implementaciones

- La mayoría de los sistemas implementan los protocolos de red como parte de ellos

Protocolos, APIs e Implementaciones

- La mayoría de los sistemas implementan los protocolos de red como parte de ellos
- Estos sistemas operativos proveen una interfaz para interactuar con la red. La llamamos API

Protocolos, APIs e Implementaciones

- La mayoría de los sistemas implementan los protocolos de red como parte de ellos
- Estos sistemas operativos proveen una interfaz para interactuar con la red. La llamamos API
- Un caso particular, la interfaz de **sockets** que originalmente eran parte de Unix (distribución de Berkley).

Protocolos, APIs e Implementaciones

- La mayoría de los sistemas implementan los protocolos de red como parte de ellos
- Estos sistemas operativos proveen una interfaz para interactuar con la red. La llamamos API
- Un caso particular, la interfaz de **sockets** que originalmente eran parte de Unix (distribución de Berkley).

Protocolos, APIs e Implementaciones

- La mayoría de los sistemas implementan los protocolos de red como parte de ellos
- Estos sistemas operativos proveen una interfaz para interactuar con la red. La llamamos API
- Un caso particular, la interfaz de **sockets** que originalmente eran parte de Unix (distribución de Berkley). La misma fue adoptada en la mayoría de los sistemas operativos hoy en día.

Importante!

Cada protocolo de red provee una serie de *servicios* y la API provee la *sintaxis* con la cual dichos servicios pueden ser invocados. La implementación, por otro lado, mapea las operaciones definidas por la API al conjunto de servicios abstractos del protocolo.

Sockets

- Es una de las principales abstracciones de la API.

Sockets

- Es una de las principales abstracciones de la API.
- Podemos pensar a un **socket** como el punto donde una aplicación se conecta a la red.

Sockets

- Es una de las principales abstracciones de la API.
- Podemos pensar a un **socket** como el punto donde una aplicación se conecta a la red.
- Tenemos operaciones para crear, enviar/recibir mensajes, etc.

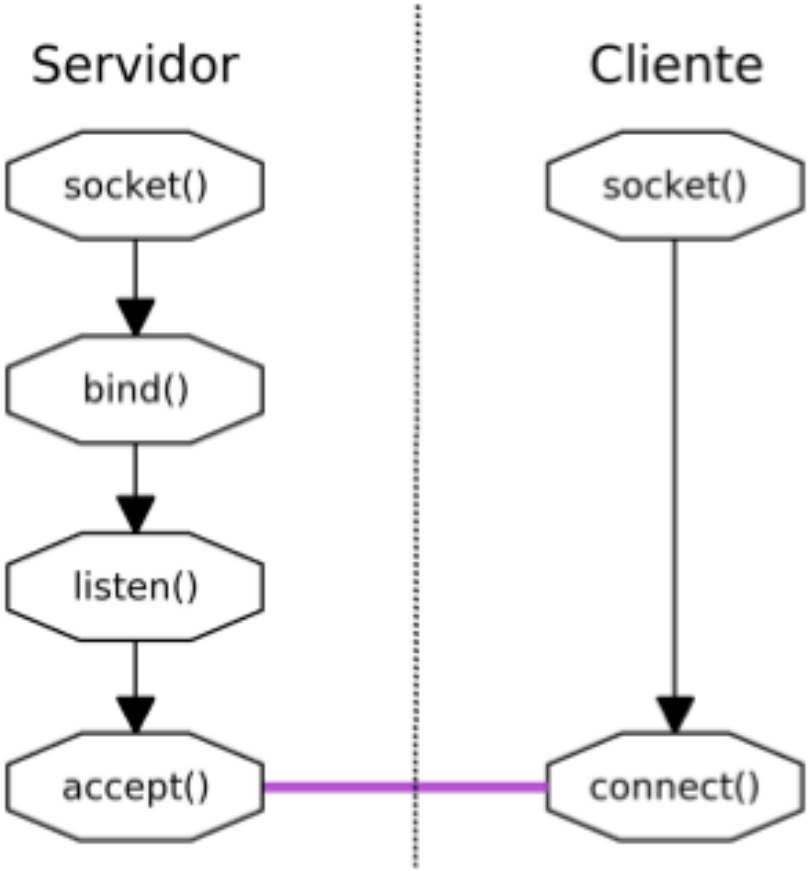
Sockets

- Es una de las principales abstracciones de la API.
- Podemos pensar a un **socket** como el punto donde una aplicación se conecta a la red.
- Tenemos operaciones para crear, enviar/recibir mensajes, etc.

Dominios

- Los sockets pueden trabajar en distintos dominios.
- En el dominio UNIX las direcciones son un nombre de archivo.
- En el dominio de Internet, las direcciones son un par <ip, puerto>
- Dos procesos que se ponen de acuerdo en tal nombre de archivo se pueden comunicar.

Funcionamiento general



Demo 1 Client, 1 Server

DEMO

Ejercicio 1: Shell Server

Realizar la comunicación entre **un** cliente y **un** servidor, que residen en el mismo host, utilizando sockets del dominio UNIX.
EL propósito de la comunicación es que el cliente pueda enviar comandos básicos, como *ls, mkdir,* etc, y que el servidor lo ejecute.
Tener en cuenta que dichos comandos pueden tener parametros. Por ejemplo: ls -l.
Sugerencia: Considerar la función **execvp**