Programación Distribuida 2013

Clase 3

1. Para la práctica: C sockets, http://www.linuxhowtos.org/C_C++/socket.htm
    1.1. Está todo en el directorio csock
    1.2. A priori, un socket no implica TCP/IP ni conexión
    1.3. man pages (tanto en la terminal Linux como en Internet):
        > man socket
        > man 7 ip (7 es la página del manual)
        > man bind
        etc.
    1.4. client.c
        1.4.1. socket(), connect(), read(), write()
        1.4.2. En el connect se proporciona el par IP/port del servidor
        1.4.3. Notar que la conexión es utilizada vía un *file descriptor* (como un arch.)
        1.4.4. Notar la relación directa de htons() con la heterogeneidad
    1.5. server.c
        1.5.1. socket(), bind(), listen(), accept(), read(), write()
        1.5.2. El socket *propio* del server no se conecta, con él se crean conexiones
        1.5.3. En el accept() se recibe un *nuevo* socket que sí está conectado
    1.6. Compilar (probablemente hay *warnings*, no se cambiaron los originales)
        > gcc -o client client.c
        > gcc -o server server.c
    1.7. Ejecutar (cada comando en una terminal diferente de una misma comp.)
        > ./server 4000
        > ./client localhost 4000
    1.8. Lo de c/s es muy discutible en este contexto
    1.9. Si *fueran* c/s se *estarían* encargando de la *transmisión* de los datos
        1.9.1. Inicialización
        1.9.2. Envío/recepción y viceversa
        1.9.3. Finalización
    1.10. Notar la “orientación” a c/s de los sockets *stream*

2. Para la práctica: Java sockets

http://docs.oracle.com/javase/tutorial/networking/sockets/

2.1. java.io.* y java.net.* son los paquetes relacionados con I/O y red/sockets

2.2. Los equivalentes a las “man pages”: descripciones de las clases

2.2.1. Búsqueda con (ej.)

Socket site:oracle.com

2.3. Client.java

2.3.1. Socket(), read(), wite()

2.3.2. Directamente se crea una conexión dando nombre/puerto

```
socketwithserver = new Socket(args[0], Integer.valueOf(args[1]));
```

2.3.3. No se puede hacer read() o write() sobre un Socket (ver descripción de la clase Socket

2.3.4. Sí se puede obtener de un Socket el InputStream y el OutputStream (*stream* de entrada y de salida del Socket respectivamente).

2.3.5. Para la práctica: relación entre DataInputStream e InputStream, DataOutputStream y OutputStream

2.4. Server.java

2.4.1. ServerSocket(), accept(), read(), write()

2.4.2. Existen *dos* clases: Socket y ServerSocket

```
serverSocket = new ServerSocket(Integer.valueOf(args[0]));
```

2.4.3. Conexión y orientación c/s: idem C sockets

2.5. Compilar

```
> javac Client.java
> javac Server.java
```

2.6. Ejecutar (cada comando en una terminal diferente de una misma comp.)

```
> java Server 4000
> java Client localhost 4000
```