Sockets: The Hard Way

Introduction to Middleware

Vojtěch Horký Petr Tůma

Department of Distributed and Dependable Systems
Faculty of Mathematics and Physics
Charles University

2010 – 2022

Outline

1. Berkeley Socket Interface
2. Assignment Part I
3. Assignment Part II
4. Marshalling Implementation
5. Assignment Part III

Interface Overview

Socket

An abstraction representing a (network) communication channel.
Both stream oriented and message oriented channels.
Spectrum of supported protocols.

Stream Oriented Channel

Socket on *client side* initiates outgoing connections.
Socket on *server side* waits for incoming connections.
Data flows in both directions after connection established.

Message Oriented Channel

No connection established.
Sender and receiver roles symmetrical.

Examples To Play With ...

```
> git clone http://github.com/d-iii-s/teaching-introduction-middleware.git
```

C

```
> cd teaching-introduction-middleware/src/sockets-basic-server/c
> cat README.md
```

Java

```
> cd teaching-introduction-middleware/src/sockets-basic-server/java
> cat README.md
```

Python

```
> cd teaching-introduction-middleware/src/sockets-basic-server/python
> cat README.md
```

Stream Oriented Channel

Client Side Pseudocode

```
socket = CreateSocket (comms_domain, socket_type);
ConnectToServer (socket, server_address);
... Write (socket, data);
... Read (socket, data);
Shutdown (socket);
Close (socket);
```

Server Side Pseudocode

```
server_socket = CreateSocket (comms_domain, socket_type);
BindToLocalAddress (socket, address);
PermitListeningOnSocket (socket, backlog);
client_socket, client_address = AcceptIncomingConnection (socket);
... Write (client_socket, data);
... Read (client_socket, data);
Shutdown (client_socket);
Close (client_socket);
```

Outline

1. Berkeley Socket Interface
2. Assignment Part I
3. Assignment Part II
4. Marshalling Implementation
5. Assignment Part III

Assignment

Server

Implement a server that will:

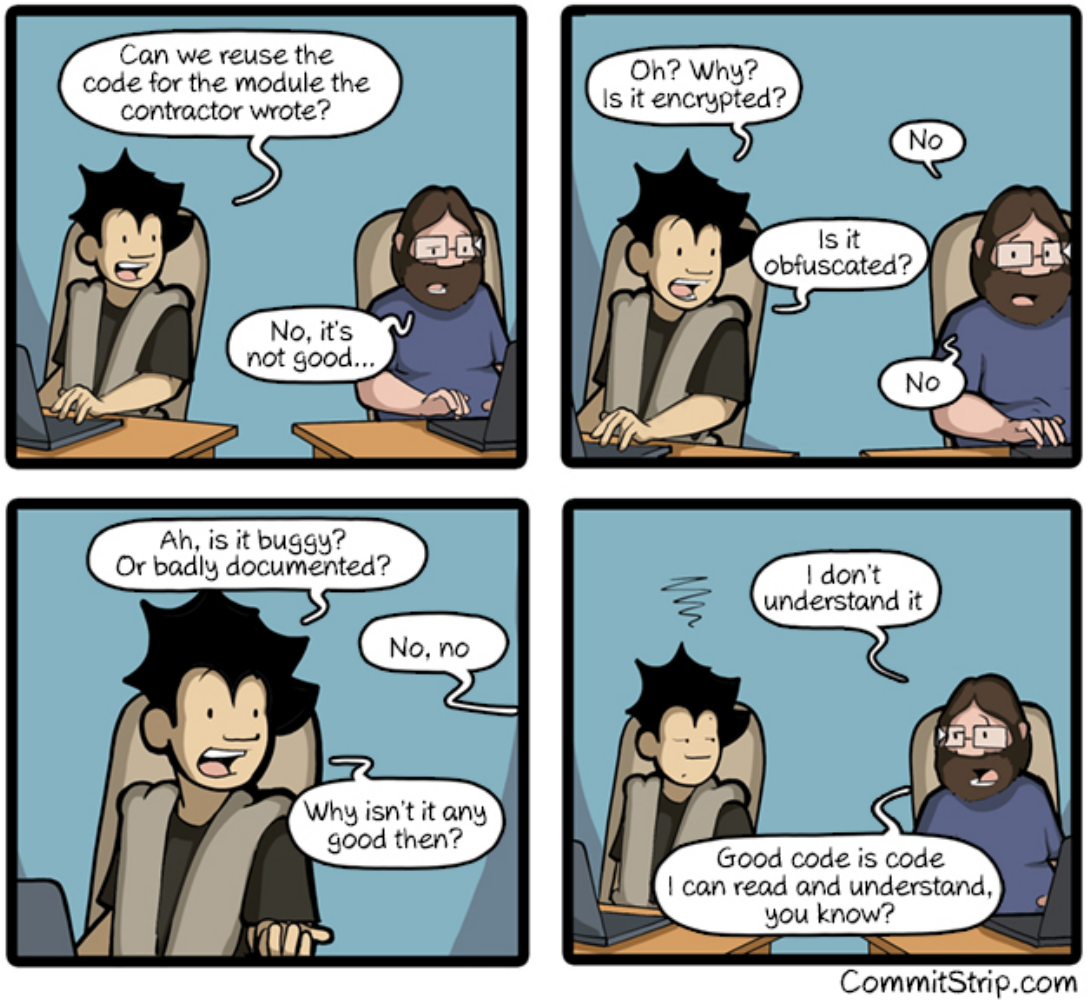
- Listen for incoming connections.
- Provide information on current time to connected clients.

Client

Implement a client that will:

- Connect to the server described above.
- Query information on current time.
- Print the time.

Code Now ...



http://www.commitstrip.com/en/2016/06/07/good-code

Show Your Code ...

Query Host Name

```
> hostname
u1-22
```

Run Screen Sharing

```
> x11vnc -viewonly
```

Outline

1. Berkeley Socket Interface
2. Assignment Part I
3. Assignment Part II
4. Marshalling Implementation
5. Assignment Part III

Assignment

Server

Implement a server that will provide information on current time.

- The server should accept a spec of what fields to return.
- Fields should be standard YYYY-MM-DD HH:MM:SS.

Client

Implement a client that will query server time:

- Pick a random combination of fields.
- Query information on current time.
- Wrap all this in a local function.
- Print the time.

C Local Function

```
/**
 * Return server time in standard structure.
 * \param result Caller allocated structure to fill.
 * \return Zero for success, non zero error code otherwise.
 */
int server_time (struct tm *result);

struct tm {
    int tm_sec;    // Seconds (0-60)
    int tm_min;    // Minutes (0-59)
    int tm_hour;   // Hours (0-23)
    int tm_mday;   // Day of the month (1-31)
    int tm_mon;    // Month (0-11)
    int tm_year;   // Year - 1900
    int tm_wday;   // Day of the week (0-6, Sunday = 0)
    int tm_yday;   // Day in the year (0-365, 1 Jan = 0)
    int tm_isdst;  // Daylight saving time
};
```

... man localtime

Java Local Function

```
/**
 * Access server time in standard structure.
 */
public interface ServerTime {
    int getSecond ();           // Gets the second-of-minute field.
    int getMinute ();           // Gets the minute-of-hour field.
    int getHour ();             // Gets the hour-of-day field.
    int getDayOfMonth ();       // Gets the day-of-month field.
    Month getMonth ();          // Gets the month-of-year field.
    int getYear ();             // Gets the year field.
    DayOfWeek getDayOfWeek ();  // Gets the day-of-week field.
    int getDayOfYear ();        // Gets the day-of-year field.
}
```

... javadoc `LocalDateTime`

Python Local Function

```
def server_time ():
    """Returns server time in datetime.datetime class."""
    ...

# Instance attributes (read-only):
#
#     datetime.year
#         Between MINYEAR and MAXYEAR inclusive.
#     datetime.month
#         Between 1 and 12 inclusive.
#     datetime.day
#         Between 1 and the number of days in the given month of the given year.
#     datetime.hour
#         In range(24).
#     datetime.minute
#         In range(60).
#     datetime.second
#         In range(60).
```

... help (datetime.datetime)

Outline

1. Berkeley Socket Interface
2. Assignment Part I
3. Assignment Part II
4. Marshalling Implementation
5. Assignment Part III

C Marshalling

Textual Stream ?

```
int sprintf (char *str, const char *format, ...);
int sscanf (const char *str, const char *format, ...);
```

Network Order Binary Stream ?

```
uint32_t htonl (uint32_t hostlong);
uint16_t htons (uint16_t hostshort);
uint32_t ntohl (uint32_t netlong);
uint16_t ntohs (uint16_t netshort);
```

Native Order Binary Stream ?

```
char buffer [1024];
int *address = (int *) &buffer [16];
*address = 1234;
```

Java Marshalling

Serialized Stream ?

```
output_stream = socket.getOutputStream ();
object_stream = new ObjectOutputStream (output_stream);
object_stream.writeInt (1234);
object_stream.writeObject (...);
```

Textual Stream ?

```
PrintWriter writer = new PrintWriter (output_stream, true);
writer.println ("...");
```

Byte Stream ?

```
ByteBuffer buffer = ByteBuffer.allocate (4);
buffer.putInt (1234);
output_stream.write (buffer.array ());
```

Python Marshalling

Pickled Stream ?

```
import pickle
with socket.makefile () as file_object:
    pickle.dump (..., file_object)
```

JSON Stream ?

```
import json
with socket.makefile () as file_object:
    json.dump (..., file_object)
```

YAML Stream ?

```
import yaml
with socket.makefile () as file_object:
    yaml.dump (..., file_object)
```

Python Marshalling

Byte Stream ?

```
data = 1234
socket.send (data.to_bytes (4, 'little'))
```

Byte Stream ?

```
from struct import *
data = pack ('bhiq', 1, 2, 3, 4)
socket.send (data)
```

Outline

1. Berkeley Socket Interface
2. Assignment Part I
3. Assignment Part II
4. Marshalling Implementation
5. Assignment Part III

Assignment

Interoperability

Implement compatible clients and servers in two languages.

Performance

Measure the performance of your implementation.

Experiment Design

Stick to the following, or provide arguments for why not:

- Random field mix, each field with probability 1/2.
- Measure at least two minutes long traffic.
- Report average invocation throughput.
- No printing during measurement.

Measuring Time

C++

```
#include <time.h>
#include <stdint.h>
struct timespec time;
clock_gettime (CLOCK_MONOTONIC_RAW, &time);
uint64_t nanoseconds =
    (uint64_t) time.tv_sec * 1000000000 +
    (uint64_t) time.tv_nsec;
```

Java

```
long nanoseconds = System.nanoTime ();
```

Python

```
import time
nanoseconds = time.clock_gettime (time.CLOCK_MONOTONIC_RAW) * 1000000000
```

Submission

GitLab

Use your personal GitLab repository under `https://gitlab.mff.cuni.cz/teaching/nswi163/2022`.

Requirements

- Use the assignment subdirectory.
- Write brief report in `SOLUTION.md`.
- Include build scripts with instructions.
- Do not commit binaries or temporary build artifacts.
- Tag your solution with `task-01` and push the tag.