Definition of the term

Socket Primitive

In Distributed Systems

Florian Willich
Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences Berlin
Course: Distributed Systems
Lecturer: Prof. Dr. Christin Schmidt

July 6, 2015

Abstract

Abstract...

Contents

1 Introduction

During my work on a technical report I came across the term *Socket Primitive* which was a heavily used term when describing operations with the Transmission Control Protocol / Internet Protocol (TCP/IP) (Tanenbaum & Steen, 2007, p. 141, ch. 4.3.1):

> [...] A socket forms an abstraction over the actual communication end point that is used by the local operating system for a specific transport protocol. In the following text, we concentrate on the socket primitives for TCP, which are shown in Fig. 4-14. [...]

Inevitably the question came up what exactly the term *Socket Primitive* defines? With this technical report, I will first clarify what a *socket* is and what a *primitive* means in computer science. Furthermore, I will describe the term *socket primitive* with examples and more detail and define the term.

Please keep in mind that although I am writing in a more abstract and general way about *socket primitives*, it is always considered to operate with a UNIX Operating System (OS).

1.1 Socket

A socket is a communication end point of a computer system. Whenever two computer systems shall communicate with each other, a connection between those systems has to be established by using interconnected sockets (Tanenbaum & Wetherall, 2011, p. 553, ch. 6.5.2).

A socket is always considered to be used associated with a protocol that implements a model of communication. Inter-Process Communication (IPC) will only be successful when both processes use the same protocol. Common protocols are the User Datagram Protocol (UDP) or TCP/IP for example. The used protocol determines what low-level mechanism is used to transmit and receive data (Loosemore *et al.*, 2015, p. 427 - 427, ch. 16).

1.2 Primitive

1.2.1 Meaning in Linguistic Science

The *Oxford Advanced Learners Dictionary* has no definition of the word *primitive* that fits for the use in computer science (A S Hornby, 2005, p. 1197). Nevertheless, the online version `http://www.oxfordlearnersdictionaries.com` offers an additional word origin:

> late Middle English (in the sense 'original, not derivative'): from Old French primitif, -ive, from Latin primitivus 'first of its kind', from primus 'first'.

Source URL: `http://www.oxfordlearnersdictionaries.com/definition/english/primitive_1?q=primitive`

The important part is the late Middle English origin in the sense 'original, not derivative'. In computer science we could understand the word *primitive* to represent the lowest level representation on the current abstraction level: Original, not derivative.

1.2.2 Meaning in Computer Science

In computer science, the word *primitive* is often used to name an instruction which represents a self-contained unit on the current abstraction level. This also means that there is probably no need for further description of what this instruction is composed of.

For example: The GNU C Reference Manual uses the term *Primitive Data Types* to describe the built-in data types in the programming language C (Rothwell & Youngman, 2015, p. 8, ch. 2). The following table shows a selection of primitive data types of a C language, their characteristics and the memory allocation on 64-bit systems. Please see Appendix A for the implementation:

Primitive	Stores	Memory Allocation
char	ASCII characters	1 byte
int	integers	4 byte
float	floating point real numbers	4 byte
double	double precision floating point real numbers	8 byte

As long as a programmer knows what each built-in *Primitive Data Type* represents and how to use it, he must not necessarily know what assembly code the compiler is generating to write code in a high level language. On the contrary, a compiler builder has to.

2 TCP/IP Socket Primitives

As already mentioned above (chapter 1.1) a socket always operates with a associated protocol. For the programmer of a high level programming language, the protocol specifies the interface to operate with the socket.

The following table lists the most common TCP/IP socket primitives with a short description (Anupama, 2007):

Primitive	Service description
Socket	creates a new socket e.g. communication end point
Bind	associates a local address with a socket
Listen	allows to accept new incoming connections to a socket
Accept	blocks until a connection request
Connect	connects to a socket
Send	sends a message to a socket
Receive	reads a message from a socket
Close	aborts a connection

Figure 2 describes the TCP/IP socket primitives (Tanenbaum & Steen, 2007, p. 142, ch. 4.3.1).

To demonstrate that a socket primitive can either be a self-contained operation for a programmer who uses a library but not for the implementer of the library, I choose the send primitive for example:

```
ssize_t send(   int socket,
                const void *buffer,
                size_t size,
                int flags)
```

3 Definition

> *A socket primitive is a socket related function that is provided by various standard libraries for various high-level programming languages. It provides an interface for calling operating system specific instructions. Therefore, a 'socket primitive' is called primitive because it is the lowest level interface for a programmer who is writing in a high level programming language to operate with network protocols.*

A Primitive Data Types in C++

```
1  #include <iostream>
2
3  /**
4   * Prints the size of the transferred value in bytes as follows:
5   *
6   * Size of 'type_name': X byte
7   *
8   * Where X is the size that is returned by the function sizeof(T).
9   *
10  * @param   type_name        The name of the type.
11  */
12 template <typename T>
13 void print_size(std::string type_name)
14 {
15     std::cout << "Size of " << type_name << ": " << sizeof(T) << "
        byte" << std::endl;
16 }
17
18 /**
19  * Prints out the following integer types:
20  *
21  * - signed char
22  * - unsigned char
23  * - char
24  * - short
25  * - short int
26  * - unsigned short int
27  * - int
```

```
 * - unsigned int
 * - long int
 * - unsigned long int
 * - long long int
 * - unsigned long long int
 */
void print_integer_types()
{
    print_size<signed char>            ("signed char");
    print_size<unsigned char>          ("unsigned char");
    print_size<char>                   ("char");
    print_size<short>                  ("short");
    print_size<short int>              ("short int");
    print_size<unsigned short int>     ("unsigned short int");
    print_size<int>                    ("int");
    print_size<unsigned int>           ("unsigned int");
    print_size<long int>               ("long int");
    print_size<unsigned long int>      ("unsigned long int");
    print_size<long long int>          ("long long int");
    print_size<unsigned long long int> ("unsigned long long int");
}

/**
 * Prints out the following real number types:
 *
 * - float
 * - double
 * - long double
 */
void print_real_number_types()
{
    print_size<float>       ("float");
    print_size<double>      ("double");
    print_size<long double> ("long double");
}

/**
 * This application prints all listed primitive data types of the
 * GNU C Reference Manual [1] excluding the complex number types
 * with information on their allocated memory:
 *
 * Integer Types:
 * - signed char
 * - unsigned char
 * - char
 * - short
 * - short int
 * - unsigned short int
 * - int
 * - unsigned int
 * - long int
 * - unsigned long int
 * - long long int
 * - unsigned long long int
 *
 * Real Number Types:
 * - float
```

```
 * - double
 * - long double
 *
 * [1] Rothwell, Trevis, & Youngman, James. 2015. The GNU C
     Reference Manual.
 * http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.pdf . Free
     Software
 * Foundation, Inc. [Online. Accessed 1st July 2015].
 *
 * @author Florian Willich
 */
int main()
{
    print_integer_types();
    print_real_number_types();

    return 0;
};
```

Acronyms

ASCII American Standard Code for Information Interchange. 2

OS Operating System. 1

TCP/IP Transmission Control Protocol / Internet Protocol. 1, 2, 3

UDP User Datagram Protocol. 1

References

A S Hornby. 2005. *Oxford Advanced Learner's Dictionary*. 7th edn. Oxford University Press, Cornelsen. ISBN 0-19-431655-6.

Anupama, Bindu. 2007. *Know your TCP system call sequences*. `http://www.ibm.com/developerworks/aix/library/au-tcpsystemcalls/au-tcpsystemcalls-pdf.pdf`. IBM Corporation. [Online. Accessed 30th June 2015].

Loosemore, Sandra, Stallman, Richard M., McGrath, Roland, Oram, Andrew, & Drepper, Ulrich. 2015. *The GNU C Library Reference Manual*. `http://www.gnu.org/software/libc/manual/pdf/libc.pdf`. Free Software Foundation, Inc. [Online. Accessed 1st July 2015].

Rothwell, Trevis, & Youngman, James. 2015. *The GNU C Reference Manual*. `http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.pdf`. Free Software Foundation, Inc. [Online. Accessed 1st July 2015].

Tanenbaum, Andrew S., & Steen, Maarten Van. 2007. *Distributed Systems: Principles and Paradigms*. 2nd edn. Pearson Prentice Hall. ISBN 0-13-239227-5.

Tanenbaum, Andrew S., & Wetherall, David J. 2011. *Computer Networks*. 5th edn. Prentice Hall. ISBN 978-0132126953.

This document was written with LaTeX
Typeface: Open Sans by Steve Matteson.