

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 4, 2015

## **Abstract**

Abstract...

## Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	SOCKET . . . . .	1
1.2	PRIMITIVE . . . . .	1
1.2.1	MEANING IN LINGUISTIC SCIENCE . . . . .	1
1.2.2	MEANING IN COMPUTER SCIENCE . . . . .	2
<b>2</b>	<b>TCP/IP SOCKET PRIMITIVES</b>	<b>2</b>
<b>3</b>	<b>DEFINITION</b>	<b>3</b>
<b>A</b>	<b>Primitive Data Types in C++</b>	<b>3</b>
	<b>References</b>	<b>6</b>

# 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 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?

It appears to be that either the term *Socket Primitive* is used when

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 socket (Tanenbaum & Wetherall, 2011, p. 553, ch. 6.5.2).

A socket is always considered to use associated with a protocol that implements a model of communication (Loosemore *et al.*, 2015, p. 427 - 428, ch. 16). Common protocols are the User Datagram Protocol (UDP) or TCP/IP for example.

## 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](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'.

### 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)

### 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) << "
16         byte" << std::endl;
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
28 * - unsigned int
29 * - long int
30 * - unsigned long int
31 * - long long int
32 * - unsigned long long int
33 */
34 void print_integer_types()
35 {
36     print_size<signed char>        ("signed char");
37     print_size<unsigned char>      ("unsigned char");
38     print_size<char>               ("char");
39     print_size<short>              ("short");
40     print_size<short int>          ("short int");
41     print_size<unsigned short int> ("unsigned short int");
42     print_size<int>                ("int");
```

```

43     print_size<unsigned int>          ("unsigned int");
44     print_size<long int>              ("long int");
45     print_size<unsigned long int>     ("unsigned long int");
46     print_size<long long int>         ("long long int");
47     print_size<unsigned long long int> ("unsigned long long int");
48 }
49
50 /**
51  * Prints out the following real number types:
52  *
53  * - float
54  * - double
55  * - long double
56  */
57 void print_real_number_types()
58 {
59     print_size<float>          ("float");
60     print_size<double>         ("double");
61     print_size<long double>    ("long double");
62 }
63
64 /**
65  * This application prints all listed primitive data types of the
66  * GNU C Reference Manual [1] excluding the complex number types
67  * with information on their allocated memory:
68  *
69  * Integer Types:
70  * - signed char
71  * - unsigned char
72  * - char
73  * - short
74  * - short int
75  * - unsigned short int
76  * - int
77  * - unsigned int
78  * - long int
79  * - unsigned long int
80  * - long long int
81  * - unsigned long long int
82  *
83  * Real Number Types:
84  * - float
85  * - double
86  * - long double
87  *
88  * [1] Rothwell, Trevis, & Youngman, James. 2015. The GNU C
      Reference Manual.
89  * http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.pdf . Free
      Software
90  * Foundation, Inc. [Online. Accessed 1st July 2015].
91  *
92  * @author Florian Willich
93  */
94 int main()
95 {
96     print_integer_types();
97     print_real_number_types();

```

```
98  
99     return 0;  
100 };
```

## Acronyms

**ASCII** American Standard Code for Information Interchange. 2

**OS** Operating System. 1

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

**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.