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 1, 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?

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 has to be established and therefore a connection between the local sockets has to be established (Tanenbaum & Wetherall, 2011, p. 553, ch. 6.5.2).

1.2 Primitive

In computer science, the word *primitive* is often used to name an instruction which represents a self-contained unit. Whenever the word *primitive* is used, the most basic instruction on the current abstraction level is ment. This also means that there will probably be no 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 implementation details:

Primitive	Characteristic	Memory Allocation
char	stores ASCII characters	1 byte
int	stores integers	4 byte
float	stores floating point real numbers	4 byte
double	stores 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.

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`

2 TCP/IP Socket Primitives

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++

```cpp
#include <iostream>

template <typename T>
void print_size(T type, std::string name)
{
    std::cout << "Size of " << name << ": " << sizeof(type) << " byte" << std::endl;
}

int main() {

    char    character { 'a' };
    int     integer_standard { 1 };
    int64_t integer_64 { 1 };
    float   floating_point { 1.0f };
    double  floating_point_double_precision { 1.0 };

    print_size<char>(character, "character (char)");
    print_size<int>(integer_standard, "standard integer (int)");
    print_size<int64_t >(integer_64, "64-bit integer (int64_t)");
    print_size<float>(floating_point, "floating point (float)");
    print_size<double>(floating_point_double_precision, "floating point with double precision (double)");

    return 0;
};
```

Acronyms

ASCII American Standard Code for Information Interchange. 1

OS Operating System. 1

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

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].

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.