ATOM TO BYTE

ALL ABOUT SOFTWARES

Introduction

Computer software, or simply software, is a collection of data or computer instructions that tell the computer how to work. In computer science and software engineering, computer software is all information processed by computer systems, programs and data. Software includes computer programs, libraries and related non-executable data, such as online documentation or digital media. In vague terms, let's understand the notion of software.

The computer system is divided into two parts – the physical part that is the hardware, and the logical part, that comprises of software. If we are asked to think about a computer system, we would usually imagine a monitor, a keyboard, a mouse, maybe speakers etc. Hardware encompasses these components. These components help us to provide input to the system and take the output. These components are mostly driven by electricity and they ought to be directed to work efficiently. So, how is it achieved?

Figure 1: A Desktop

Under the hood

The language of computers is the binary language: 0 or 1. This is achieved by giving electric pulses (DC voltage) to the system. In the core of these systems, we find registers that use the logic gates to alter and/or store bits of information. A bit of information is nothing but a state of the pulse applied – 0 for off, and 1 for on. Many such registers come together to form the system storage. But we can't communicate with these components directly – there is a huge gap between what we say and what a system understands. So we need a translator to help communicate with the system. Each character that we type through the keyboard is mapped with a fixed sequence of these 0 and 1 bits. 8 such bits form a byte. So a byte can have 2^8 possible combinations of 0s and 1s that sums up to 256. Now just imagine, with a storage power of 100 GB *, how many characters can one store!

So now, we use an assembler to translate our commands to the binary form. An assembler is just a name given to the translator because it translates the assembly language to the binary one. Even the assembly language is not so easy to use – due to the involvement of basic commands only and more importantly, lack of a structure.

So we started creating languages that are human understandable, which are formally called as High level languages. Now we needed an interface to convert these languages into assembly languages. So this gave rise to compilers and interpreters. Most of the languages that we use today like C, C++, JAVA, and Python etc. are high level languages and they all need a compiler of their own to act as an interface between the system and user.

* 1 GB = 1 Gigabyte, it is 1024 x 1024 x 1024 x 8 bits i.e. $2^{8589934592}$ combinations

History

An outline (algorithm) for what would have been the first piece of software was written by Ada Lovelace in the 19th century, for the planned Analytical Engine. She created proofs to show how the engine would calculate Bernoulli Numbers. Because of the proofs and the algorithm, she is considered the first computer programmer. The first theory about software prior to the creation of computers as we know them today was proposed by Alan Turing in his 1935 essay On Computable Numbers, with an Application to the Entscheidung's problem (decision problem).This eventually led to the creation of the academic fields of computer science and software engineering; both fields study software and its creation. Computer science is the theoretical study of computer and software (Turing's essay is an example of computer science), whereas software engineering is the application of engineering and development of software.

However, prior to 1946, software was not yet the programs stored in the memory of stored-program digital computers, as we now understand it. The first electronic computing devices were instead rewired in order to "reprogram" them. In 2000, Fred Shapiro, a librarian at the Yale Law School, published a letter revealing that John Wilder Tukey's 1958 paper "The Teaching of Concrete Mathematics" contained the earliest known usage of the term "software" found in a search of JSTOR's electronic archives, predating the OED's citation by two years. This led many to credit Tukey with coining the term, particularly in obituaries published that same year, although Tukey never claimed credit for any such coinage. In 1995, Paul Niquette claimed he had originally coined the term in October 1953, although he could not find any documents supporting his claim. The earliest known publication of the term "software" in an engineering context was in August 1953 by Richard R. Carhart, in a Rand Corporation Research Memorandum.

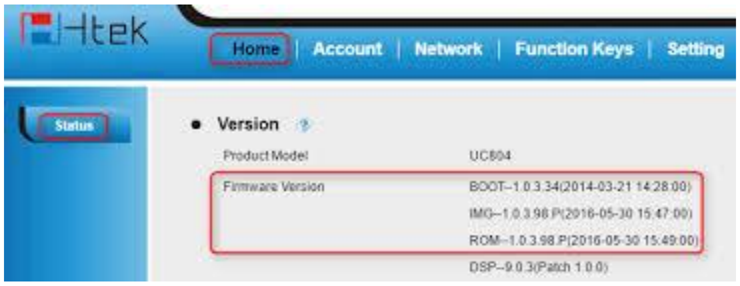
Classification

Software can be classified on many grounds. But they can be put to three main groups –

- System Software
- Application Software
- Firmware

Application software

System Software



Version

Product Model	UC804
Firmware Version	BOOT--1.0.3.34(2014-03-21 14:28:00)
	IMG--1.0.3.98.P(2016-05-30 15:47:00)
	ROM--1.0.3.98.P(2016-05-30 15:49:00)
	DSP--9.0.3(Patch 1.0.0)

Firmware

System Software

The system software is a collection of programs designed to operate, control, and extend the processing capabilities of the computer itself. System software is generally prepared by the computer manufacturers. These software products comprise of programs written in low-level languages, which interact with the hardware at a very basic level. System software serves as the interface between the hardware and the end users.

Some examples of system software are Operating System, Compilers, Interpreter, Assemblers, etc. The operating system (prominent examples being Microsoft Windows, mac OS, Linux, and z/OS), allows the parts of a computer to work together by performing tasks like transferring data between memory and disks or rendering output onto a display device. It provides a platform (hardware abstraction layer) to run high-level system software and application software.

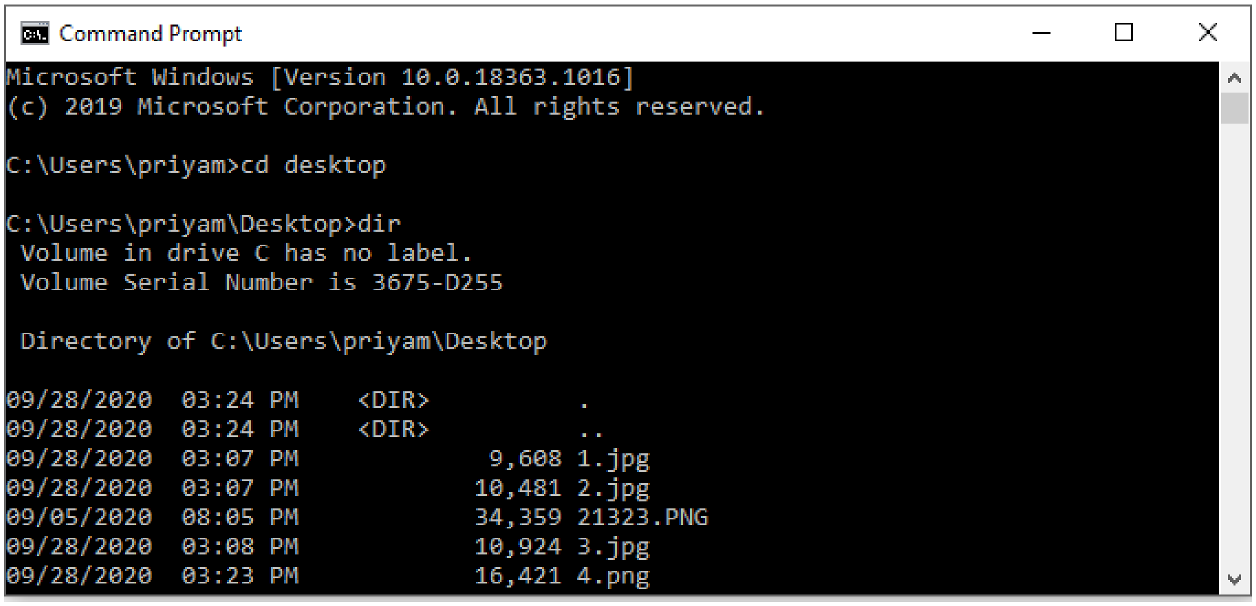
A kernel is the core part of the operating system that defines an API for applications programs (including some system software) and an interface to device drivers.

Device drivers, including also computer BIOS and device firmware, provide basic functionality to operate and control the hardware connected to or built into the computer.

A user interface allows users to interact with a computer. Either a command-line interface (CLI) or, since the 1980s a graphical user interface (GUI). Since this is the part of the operating system the user directly interacts with, it may be considered an application and therefore not system software.

Here is a list of some of the most prominent features of system software −

- Close to the system
- Fast in speed
- Difficult to design
- Difficult to understand
- Less interactive
- Smaller in size
- Difficult to manipulate
- Generally written in low-level language



```
Microsoft Windows [Version 10.0.18363.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\priyam>cd desktop

C:\Users\priyam\Desktop>dir
 Volume in drive C has no label.
 Volume Serial Number is 3675-D255

 Directory of C:\Users\priyam\Desktop

09/28/2020  03:24 PM    <DIR>          .
09/28/2020  03:24 PM    <DIR>          ..
09/28/2020  03:07 PM             9,608 1.jpg
09/28/2020  03:07 PM            10,481 2.jpg
09/05/2020  08:05 PM            34,359 21323.PNG
09/28/2020  03:08 PM            10,924 3.jpg
09/28/2020  03:23 PM            16,421 4.png
```

Command prompt (DOS)

Application Software

Application software (app for short) is a program or group of programs designed for end users. Examples of an application include a word processor, a spreadsheet, an accounting application, a web browser, an email client, a media player, a file viewer, simulators, a console game or a photo editor. The collective noun application software refers to all applications collectively. This contrasts with system software, which is mainly involved with running the computer.

Applications may be bundled with the computer and its system software or published separately, and may be coded as proprietary, open-source or university projects. Apps built for mobile platforms are called mobile apps. There are many different and alternative ways in order to classify application software. By the legal point of view, application software is mainly classified with a black box approach, in relation to the rights of its final end-users or subscribers (with eventual intermediate and tiered subscription levels). Software applications are also classified in respect of the programming language in which the source code is written or executed, and respect of their purpose and outputs.

They are further classified on grounds such as –

- Properties and Rights
- Coding language
- Purpose and Output

By property and use rights

Application software is usually distinguished among two main classes: closed source vs. open source software applications, and among free or proprietary software applications. Proprietary software is placed under the exclusive copyright and a software license grants limited usage rights. The open-closed principle states that software may be "open only for extension, but not for modification". Such applications can only get add-on by third-parties. Free and open-source software shall be run, distributed, sold or extended for any purpose, and -being open- shall be modified or reversed in the same way.

FOSS software applications released under a free license may be perpetual and also royalty-free. Perhaps, the owner, the holder or third-party enforcer of any right (copyright, trademark or patent) are entitled to add exceptions, limitations, time decays or expiring dates to the license terms of use.

Public-domain software is a type of FOSS, which is royalty-free and openly or reservedly can be run, distributed, modified, reversed, republished or created in derivative works without any copyright attribution and therefore revocation. It can even be sold, but without transferring the public domain property to other single subjects. Public-domain SW can be released under a legal statement, which enforces those terms and conditions for an indefinite duration (for a lifetime, or forever).

E.g. MS Office (Proprietary) vs. Open Office (Open Source)

By coding language

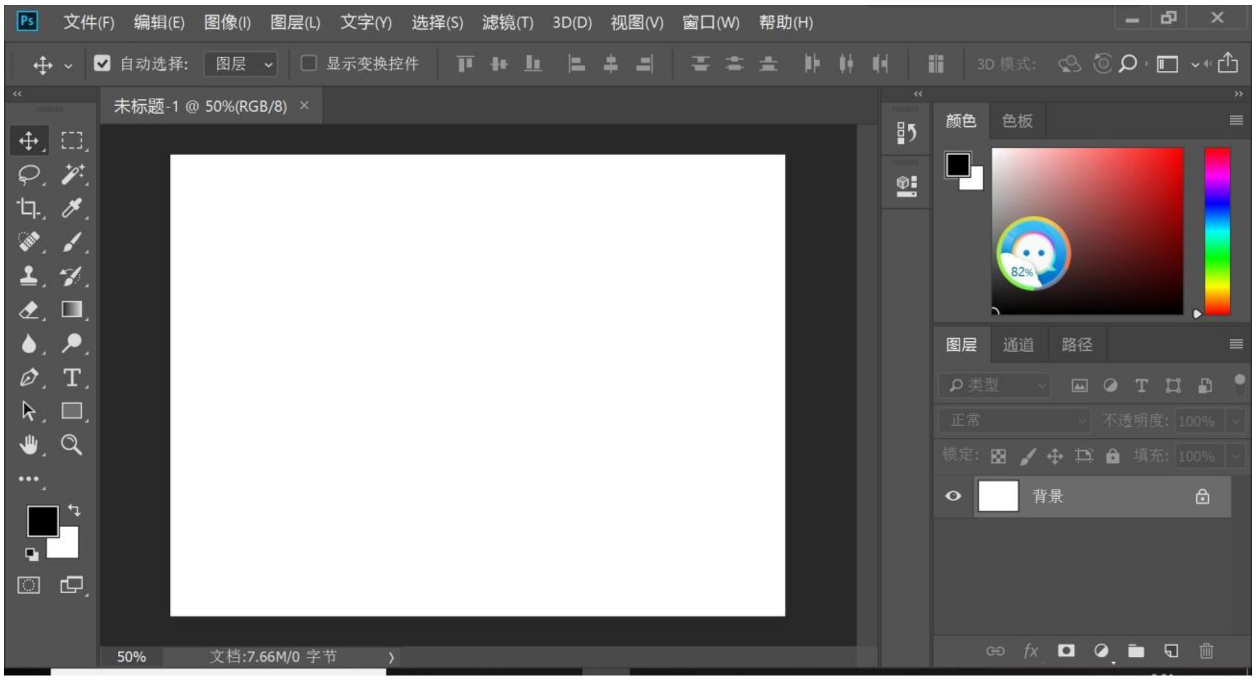
Since the development and near-universal adoption of the web, an important distinction that has emerged, has been between web applications — written with HTML, JavaScript and other web-native technologies and typically requiring one to be online and running a web browser, and the more traditional native applications written in whatever languages are available for one's particular type of computer. There has been a contentious debate in the computing community regarding web applications replacing native applications for many purposes, especially on mobile devices such as smartphones and tablets. Web apps have indeed greatly increased in popularity for some uses, but the advantages of applications make them unlikely to disappear soon, if ever. Furthermore, the two can be complementary, and even integrated.

By purpose and output

Application software can also be seen as being either horizontal or vertical. Horizontal applications are more popular and widespread, because they are general purpose, for example word processors or databases. Vertical applications are niche products, designed for a particular type of industry or business, or department within an organization. Integrated suites of software will try to handle every specific aspect possible of, for example, manufacturing or banking worker, or accounting, or customer service.

Features of application software are as follows −

- Close to the user
- Easy to design
- More interactive
- Slow in speed
- Generally written in high-level language
- Easy to understand
- Easy to manipulate and use
- Bigger in size and requires large storage space



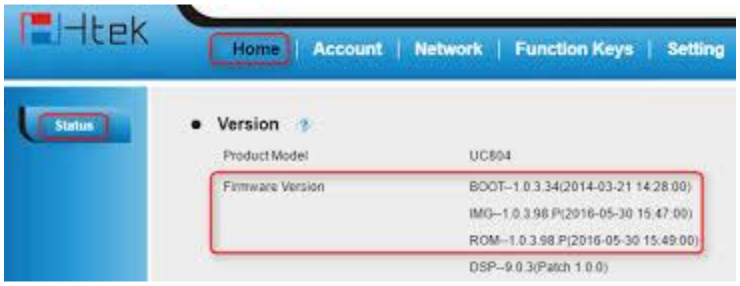
Adobe Photoshop (Application software)

Firmware

Firmware is a software program permanently etched into a hardware device such as a keyboards, hard drive, BIOS, or video cards. It is programmed to give permanent instructions to communicate with other devices and perform functions like basic input/output tasks. Firmware is typically stored in the flash ROM (read only memory) of a hardware device. It can be erased and rewritten.

Firmware was originally designed for high level software and could be changed without having to exchange the hardware for a newer device. Firmware also retains the basic instructions for hardware devices that make them operative. Without firmware, a hardware device would be non-functional.

Originally, firmware had read-only memory (ROM) and programmable read-only memory (PROM). It was designed to be permanent. Eventually PROM chips could be updated and were called erasable programmable read-only memory (EPROM). But EPROM was expensive, time consuming to update and challenging to use. Firmware eventually evolved from ROM to flash memory firmware; thus, it became easier to update and user friendly.



● Version

Product Model	UC804
Firmware Version	BOOT–1.0.3.34(2014-03-21 14:28:00)
	IMG–1.0.3.98.P(2016-05-30 15:47:00)
	ROM–1.0.3.98.P(2016-05-30 15:49:00)
	DSP–9.0.3(Patch 1.0.0)

Firmware

Levels of firmware

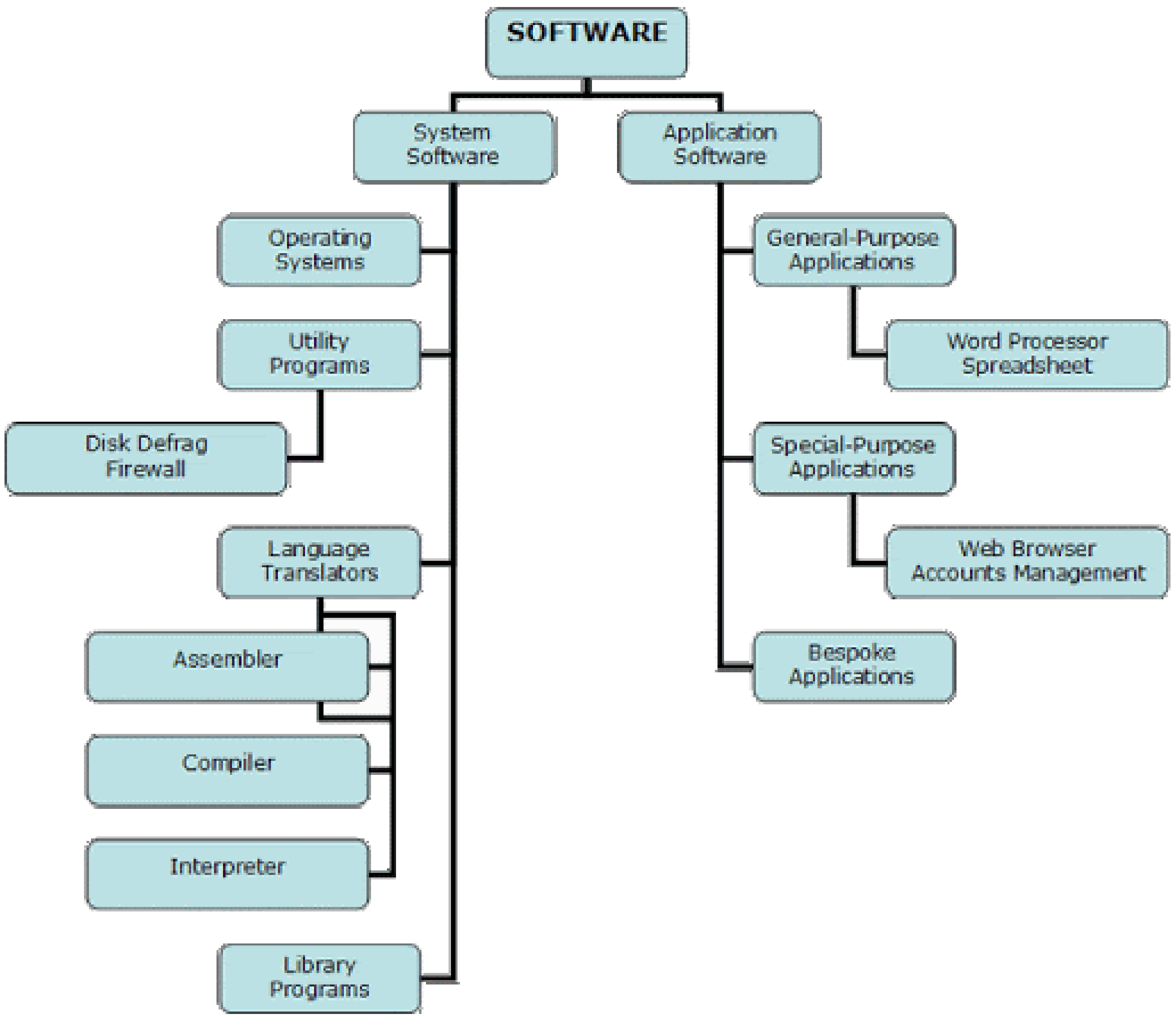
Low Level Firmware: This is found in ROM, OTP/PROM and PLA structures. Low level firmware is often read-only memory and cannot be changed or updated. It is sometimes referred to as hardware.

High Level Firmware: This is used in flash memory for updates that is often considered as software.

Subsystems: These have their own fixed microcode embedded in flash chips, CPUs and LCD units. A subsystem is usually considered part of the hardware device as well as high level firmware.

BIOS, modems and video cards are usually easy to update. But firmware in storage devices usually gets overlooked; there are no standardized systems for updating firmware. Fortunately, storage devices do not need to be updated often.

Mind Map


SOFTWARE
System Software
Application Software
Operating Systems
Utility Programs
Disk Defrag Firewall
Language Translators
Assembler
Compiler
Interpreter
Library Programs
General-Purpose Applications
Word Processor Spreadsheet
Special-Purpose Applications
Web Browser Accounts Management
Bespoke Applications

Acknowledgements-

- Wikipedia
- Techopedia
- W3schools

--x---