# Learning to program with F#

Jon Sporring

Department of Computer Science,
University of Copenhagen

September 3, 2017

# Contents

# A | The console in Windows, MacOS X, and Linux

Almost all popular operating systems are accessed through a user-friendly *graphical user interface* (*GUI*) that is designed to make typical tasks easy to learn to solve. As a computer programmer, you often need to access some of the functionalities of the computer, which, unfortunately, are sometimes complicated by this particular graphical user interface. The *console*, also called the *terminal* and the *Windows command line*, is the right hand of a programmer. The console is a simple program that allows you to complete text commands. Almost all the tasks that can be done with the graphical user interface can be done in the terminal and vice versa. Using the console, you will benefit from the terminal's direct control of the programs we write, and in your education, you will benefit from the fast and raw information you get through the console.

- graphical user interface
- GUI
- console
- terminal
- Windows command line

## A.1 The basics

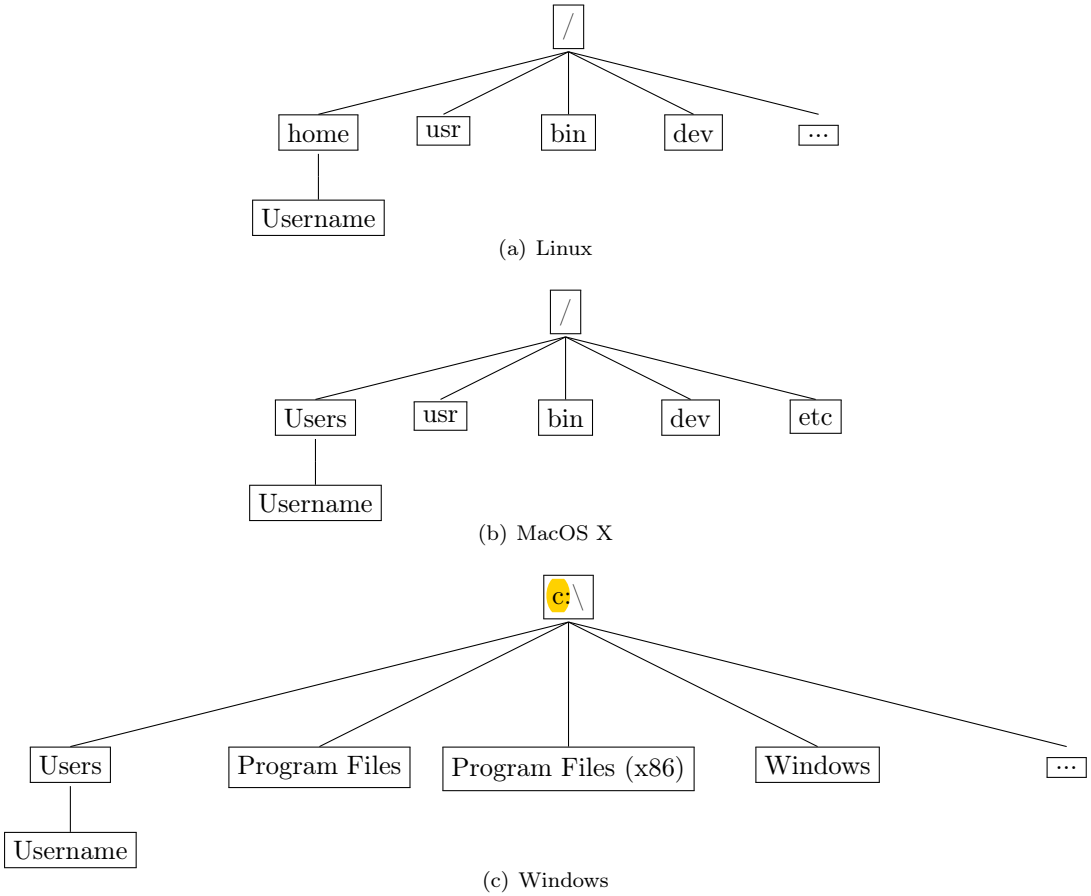
When you open a *directory* or *folder* in your preferred operating system, the directory will have a location in the file system, whether from the terminal or through the operating system's graphical user interface. The console will almost always be associated with a particular directory or folder in the file system, and it is said that it is the directory that the console is in. The exact structure of file systems varies between Linux, MaxOS X and Windows, but common is that it is a hierarchical structure. This is illustrated in Figure A.1.

- directory
- folder

There are many predefined console commands, available in the console, and you can make your own. In the following, we will review the most important commands in the three different operating systems. These are summarized in Table A.1.

## A.2 Windows

In this section we will discuss the commands summarized in Table A.1. Windows 7 and earlier versions: To open the console, press `Start->Run` in the lower left corner, and then type `cmd` In the box. Alternatively you can type `Windows-key + R`. Now you should open a terminal window with a prompt showing something like Listing A.1.




(a) Linux

(b) MacOS X

(c) Windows

Figure A.1: The top file hierarchy levels of common operating systems.

| Windows | MacOS X/Linux | Description |
|---|---|---|
| `dir` | `ls` | Show content of present directory. |
| `cd <dir>` | `cd <dir>` | Change present directory to `<dir>`. |
| `mkdir <dir>` | `mkdir <dir>` | create directory `<dir>`. |
| `rmdir <dir>` | `rmdir <dir>` | delete `<dir>` (Warning: cannot be reverted). |
| `move <file> <file or dir>` | `mv <file> <file or dir>` | Move `<fil>` to `<file or dir>`. |
| `copy <file> <file>` | `cp <file> <file>` | Create a new file called `<file>` as a copy of `<file>`. |
| `del <file>` | `rm <file>` | delete `<file>` (Warning: cannot be reverted). |
| `echo <string or variable>` | `echo <string or variable>` | Write a string or content of a variable to screen. |

Table A.1: The most important console commands for Windows, MacOS X, and Linux.

**Listing A.1: The Windows console.**

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\sporring>
```

To see, which files are in the directory use *dir* as shown in Listing A.2. · dir

**Listing A.2: Directory listing with dir.**

```
C:\Users\sporring>dir
 Volume in drive C has no label.
 Volume Serial Number is 94F0-31BD

 Directory of C:\Users\sporring

30-07-2015  15:23    <DIR>          .
30-07-2015  15:23    <DIR>          ..
30-07-2015  14:27    <DIR>          Contacts
30-07-2015  14:27    <DIR>          Desktop
30-07-2015  17:40    <DIR>          Documents
30-07-2015  15:11    <DIR>          Downloads
30-07-2015  14:28    <DIR>          Favorites
30-07-2015  14:27    <DIR>          Links
30-07-2015  14:27    <DIR>          Music
30-07-2015  14:27    <DIR>          Pictures
30-07-2015  14:27    <DIR>          Saved Games
30-07-2015  17:27    <DIR>          Searches
30-07-2015  14:27    <DIR>          Videos
               0 File(s)              0 bytes
              13 Dir(s)  95.004.622.848 bytes free

C:\Users\sporring>
```

We see that there are no files and thirteen directories (DIR), the date and time of their creation in the two left columns, their size or if it is a folder, as well as their names in the right-hand column. The first two folders `.` and `..` are found in each folder and refers to this folder as well as the one above in the hierarchy. In this case, the folder `.` An alias for `C:\Users\tracking` and `..` for `C:\Users`.

Use *cd* to change directory, e.g., to `Documents` as in Listing A.3. · cd

**Listing A.3: Change directory with cd.**

```
C:\Users\sporring>cd Documents

C:\Users\sporring\Documents>
```

The directory `Documents` is that which Windows Explorer selects, when you press the `Documents` short-cut in Explorer. Note that some systems translate default filenames so that their names may be given different names in different languages in the graphical user interface as compared to the terminal. On a new computer is the `Documents` folder is typically empty, as shown in Listing A.4.

**Listing A.4: A new directory is always empty.**

```
C:\Users\sporring\Documents>dir
 Volume in drive C has no label.
 Volume Serial Number is 94F0-31BD

 Directory of C:\Users\sporring\Documents

30-07-2015  19:16    <DIR>          .
30-07-2015  19:16    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  94.656.716.800 bytes free

C:\Users\sporring\Documents>
```

You can use *mkdir* to create a new directory called, e.g., `myFolder` as illustrated in Listing A.5.

· `mkdir`

**Listing A.5: Creating a directory with `mkdir`.**

```
C:\Users\sporring\Documents>mkdir minMappe

C:\Users\sporring\Documents>dir
 Volume in drive C has no label.
 Volume Serial Number is 94F0-31BD

 Directory of C:\Users\sporring\Documents

30-07-2015  19:17    <DIR>          .
30-07-2015  19:17    <DIR>          ..
30-07-2015  19:17    <DIR>          myFolder
               0 File(s)              0 bytes
               3 Dir(s)  94.656.638.976 bytes free

C:\Users\sporring\Documents>
```

By using `dir` we inspect the result.

Files can be created, e.g., *echo* and *redirection* as demonstrated in Listing A.6.

· `echo`
· redirection

**Listing A.6: The Windows console.**

```
C:\Users\sporring\Documents>echo "Hi" > hi.txt

C:\Users\sporring\Documents>dir
 Volume in drive C has no label.
 Volume Serial Number is 94F0-31BD

 Directory of C:\Users\sporring\Documents

30-07-2015  19:18    <DIR>          .
30-07-2015  19:18    <DIR>          ..
30-07-2015  19:17    <DIR>          myFolder
30-07-2015  19:18                 8 hi.txt
               1 File(s)              8 bytes
               3 Dir(s)  94.656.634.880 bytes free

C:\Users\sporring\Documents>
```

To move the file `hi.txt` to the directory `myFolder` use *move* as shown in Listing A.7.

· move

**Listing A.7: Move a file with `move`.**

```
C:\Users\sporring\Documents>move hi.txt myFolder
        1 file(s) moved.

C:\Users\sporring\Documents>
```

Finally, use *del* to delete a file and *rmdir* to delete a directory as shown in Listing A.8.

· del
· rmdir

**Listing A.8: Delete files and directories with `del` and `rmdir`.**

```
C:\Users\sporring\Documents>cd myFolder

C:\Users\sporring\Documents\minMappe>del hi.txt

C:\Users\sporring\Documents\minMappe>cd ..

C:\Users\sporring\Documents>rmdir myFolder

C:\Users\sporring\Documents>dir
 Volume in drive C has no label.
 Volume Serial Number is 94F0-31BD

 Directory of C:\Users\sporring\Documents

30-07-2015  19:20    <DIR>          .
30-07-2015  19:20    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  94.651.142.144 bytes free

C:\Users\sporring\Documents>
```

The commands available from the console must be in its *search path*. The search path can be seen

· search path

using `echo` as shown in Listing A.9.

**Listing A.9: The Windows console.**

```
C:\Users\sporring\Documents>echo %Path%
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;
   C:\Windows\System32\WindowsPowerShell\v1.0\;"\Program
   Files\emacs-24.5\bin\"

C:\Users\sporring\Documents>
```

The path can be changed using the Control panel in the graphical user interface. In the Control panel, choose `System and Security` → `System` → `Advanced system settings` → `Environment Variables`, and in the `System variables` box, double-click on `Path` and add or remove a path from the list. The search path is a list of paths separated by ";". Beware, Windows uses the search path for many different tasks, so remove only paths that you are certain are not used for anything.

A useful feature of the console is that you can use the `tab`-key to cycle through filenames. E.g., if you write `cd` followed by a space and `tab` a couple of times, then the terminal will suggest you the available directories.

## A.3 MacOS X and Linux

MacOS X (OSX) and Linux are very similar, and both have the option of using ***bash*** as console. It is in the standard console on MacOS X and on many Linux distributions. A summary of the most important `bash` commands are shown in Table A.1. In MacOS X, you find the console by opening `Finder` and navigate to `Applications` → `Utilities` -> `Terminal`. In Linux, the console can be started by typing `Ctrl + Alt + T`. Some Linux distributions have other key-combinations such as `Super + T`.

· **bash**

Once opened, the console is shown in a window with content as shown in Listing A.10.

**Listing A.10: The Windows console.**

```
Last login: Thu Jul 30 11:52:07 on ttys000
FN11194:~ sporring$
```

"FN11194" is the name of the computer, the character ∼ is used as an alias for the user's home directory, and "sporring" is the username for the user presently logged onto the system. Use ***ls*** to see which files are present, as shown in Listing A.11.

· **ls**

**Listing A.11: Display a directory content with `ls`.**

```
FN11194:~ sporring$ ls
Applications    Documents       Library         Music           Public
Desktop         Downloads       Movies          Pictures
FN11194:~ sporring$
```

More details about the files are available by using flags to `ls` as demonstrated in Listing A.12.

**Listing A.12: Display extra information about files using flags to `ls`.**

```
FN11194:~ sporring$ ls -l
drwx------   6 sporring  staff   204 Jul 30 14:07 Applications
drwx------+ 32 sporring  staff  1088 Jul 30 14:34 Desktop
drwx------+ 76 sporring  staff  2584 Jul  2 15:53 Documents
drwx------+  4 sporring  staff   136 Jul 30 14:35 Downloads
drwx------@ 63 sporring  staff  2142 Jul 30 14:07 Library
drwx------+  3 sporring  staff   102 Jun 29 21:48 Movies
drwx------+  4 sporring  staff   136 Jul  4 17:40 Music
drwx------+  3 sporring  staff   102 Jun 29 21:48 Pictures
drwxr-xr-x+  5 sporring  staff   170 Jun 29 21:48 Public
FN11194:~ sporring$
```

The flag `-l` means long, and many other flags can be found by querying the built-in manual with `man ls`. The output is divided into columns, where the left column shows a number of codes: "d" means they are directory followed by three sets of optional "rwx" denoting whether the owner, the associated group of users and anyone can "r" - read, "w" - write, or "x" - execute the files. In this case, only the owner can any of the three. For directories, "x" means permission to enter. The second column can often be ignored, but shows how many links there are to the file or directory. Then follows the username of the owner, which in this case is `sporring`. The files are also associated with a group of users, and in this case, they all are associate with the group called `staff`. Then follows the file or directory size, the date of last change, and the file or directory name. There are always two hidden directories: `.` and `..`, where `.` is an alias for the present directory, and `..` for the directory above. Hidden files will be shown with the `-a` flag.

Use *`cd`* to change to the directory `Documents` as shown in Listing A.13. · `cd`

**Listing A.13: Change directory with `cd`.**

```
FN11194:~ sporring$ cd Documents/
FN11194:Documents sporring$
```

Note that some graphical user interfaces translate standard filenames and directories to the local language, such that navigating using the graphical user interface will reveal other files and directories, which, however, are aliases. On a new computer many directories are predefined but empty, e.g., `Documents` as shown in Listing A.14.

**Listing A.14: New directories are empty.**

```
FN11194:Documents sporring$ ls
FN11194:Documents sporring$
```

You can yourself create a new directory using *`mkdir`* as demonstrated in Listing A.15. · `mkdir`

**Listing A.15: Creating a directory using `mkdir`.**

```
FN11194:Documents sporring$  mkdir myFolder
FN11194:Documents sporring$ ls
myFolder
FN11194:tmp sporring$
```

A file can be created using *echo* and with *redirection* as shown in Listing A.16.

· echo
· redirection

**Listing A.16: Creating a directory with `echo`.**

```
FN11194:Documents sporring$ echo "hi" > hi.txt
FN11194:Documents sporring$ ls
hi.txt          myFolder
```

To move the file `hi.txt` into `myFolder` use *mv*. This is demonstrated in Listing A.17.

· mv

**Listing A.17: Moving files with `mv`.**

```
FN11194:Documents sporring$ echo mv hi.txt myFolder/
FN11194:Documents sporring$
```

To delete the file and the directory, use *rm* and *rmdir* as shown in Listing A.18.

· rm
· rmdir

**Listing A.18: The Windows console.**

```
FN11194:Documents sporring$ cd myFolder/
FN11194:myFolder sporring$ rm hi.txt
FN11194:myFolder sporring$ cd ..
FN11194:Documents sporring$ rmdir myFolder/
FN11194:Documents sporring$ ls
FN11194:Documents sporring$
```

Only commands found on the *search-path* are available in the console. The content of the search-path is seen using the `echo` command as demonstrated in Listing A.19.

· search-path

**Listing A.19: The content of the search-path.**

```
FN11194:Documents sporring$ echo $PATH
/Applications/Maple 17/:/Applications/PackageMaker.app/Contents/MacOS/:
   /Applications/MATLAB_R2014b.app/bin/:/opt/local/bin:/opt/local/sbin:
   /usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:
   /Library/TeX/texbin
FN11194:Documents sporring$
```

The search-path can be changed by editing the setup file for Bash. On MacOS X it is called `~/.profile`, and on Linux it is either `~/.bash_profile` or `~/.bashrc`. Here new paths can be added by adding the following line `export PATH="<new path>:<another new path>:$PATH"`.

A useful feature of Bash is that the console can help you write commands. E.g., if you write `fs` followed by pressing the `tab`-key, and if Mono is in the search-path, then Bash will typically respond by completing the line as `fsharp`, and by further pressing the tab-key sometimes, Bash will show the list of options, typically `fshpari` and `fsharpc`. Also, most commands have an extensive manual, which can be accessed using the `man` command. E.g., the manual for `rm` is retrieved by `man rm`.

# B | Number systems on the computer

## B.1 Binary numbers

Humans like to use the *decimal number* system for representing numbers. Decimal numbers are *base* 10 means that for a number consisting of a sequence of digits separated by a *decimal point*, where each *digit* can have values $d \in \{0, 1, 2, \ldots, 9\}$ and the weight of each digit is proportional to its place in the sequence of digits w.r.t. the decimal point, i.e., the number $357.6 = 3 \cdot 10^2 + 5 \cdot 10^1 + 7 \cdot 10^0 + 6 \cdot 10^{-1}$ or in general:

- decimal number
- base
- decimal point
- digit

$$v = \sum_{i=-m}^{n} d_i 10^i \tag{B.1}$$

The basic unit of information in almost all computers is the binary digit or *bit* for short. A *binary number* consists of a sequence of binary digits separated by a decimal point, where each digit can have values $b \in \{0, 1\}$, and the base is 2. The general equation is,

- bit
- binary number

$$v = \sum_{i=-m}^{n} b_i 2^i \tag{B.2}$$

and examples are $1011.1_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} = 11.5$. Notice that we use subscript 2 to denote a binary number, while no subscript is used for decimal numbers. The left-most bit is called the *most significant bit*, and the right-most bit is called the *least significant bit*. Due to typical organization of computer memory, 8 binary digits is called a *byte*, and 32 digits a *word*.

- most significant bit
- least significant bit
- byte
- word

Other number systems are often used, e.g., *octal numbers*, which are base 8 numbers, where each digit is $o \in \{0, 1, \ldots, 7\}$. Octals are useful short-hand for binary, since 3 binary digits maps to the set of octal digits. Likewise, *hexadecimal numbers* are base 16 with digits $h \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f\}$, such that $a_{16} = 10$, $b_{16} = 11$ and so on. Hexadecimals are convenient since 4 binary digits map directly to the set of octal digits. Thus $367 = 101101111_2 = 557_8 = 16f_{16}$. A list of the integers 0–63 is various bases is given in Table B.1.

- octal number
- hexadecimal number

## B.2 IEEE 754 floating point standard

The set of real numbers also called *reals* includes all fractions and irrational numbers. It is infinite in size both in the sense that there is no largest nor smallest number and between any 2 given numbers there are infinitely many numbers. Reals are widely used for calculation, but since any computer only has finite memory, it is impossible to represent all possible reals. Hence, any computation performed on a computer with reals must rely on approximations. *IEEE 754 double precision floating-point format* (*binary64*), known as a *double*, is a standard for representing an approximation of reals using 64 bits.

- reals
- IEEE 754 double precision floating-point format
- binary64
- double

| Dec | Bin | Oct | Hex | Dec | Bin | Oct | Hex |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 32 | 100000 | 40 | 20 |
| 1 | 1 | 1 | 1 | 33 | 100001 | 41 | 21 |
| 2 | 10 | 2 | 2 | 34 | 100010 | 42 | 22 |
| 3 | 11 | 3 | 3 | 35 | 100011 | 43 | 23 |
| 4 | 100 | 4 | 4 | 36 | 100100 | 44 | 24 |
| 5 | 101 | 5 | 5 | 37 | 100101 | 45 | 25 |
| 6 | 110 | 6 | 6 | 38 | 100110 | 46 | 26 |
| 7 | 111 | 7 | 7 | 39 | 100111 | 47 | 27 |
| 8 | 1000 | 10 | 8 | 40 | 101000 | 50 | 28 |
| 9 | 1001 | 11 | 9 | 41 | 101001 | 51 | 29 |
| 10 | 1010 | 12 | a | 42 | 101010 | 52 | 2a |
| 11 | 1011 | 13 | b | 43 | 101011 | 53 | 2b |
| 12 | 1100 | 14 | c | 44 | 101100 | 54 | 2c |
| 13 | 1101 | 15 | d | 45 | 101101 | 55 | 2d |
| 14 | 1110 | 16 | e | 46 | 101110 | 56 | 2e |
| 15 | 1111 | 17 | f | 47 | 101111 | 57 | 2f |
| 16 | 10000 | 20 | 10 | 48 | 110000 | 60 | 30 |
| 17 | 10001 | 21 | 11 | 49 | 110001 | 61 | 31 |
| 18 | 10010 | 22 | 12 | 50 | 110010 | 62 | 32 |
| 19 | 10011 | 23 | 13 | 51 | 110011 | 63 | 33 |
| 20 | 10100 | 24 | 14 | 52 | 110100 | 64 | 34 |
| 21 | 10101 | 25 | 15 | 53 | 110101 | 65 | 35 |
| 22 | 10110 | 26 | 16 | 54 | 110110 | 66 | 36 |
| 23 | 10111 | 27 | 17 | 55 | 110111 | 67 | 37 |
| 24 | 11000 | 30 | 18 | 56 | 111000 | 70 | 38 |
| 25 | 11001 | 31 | 19 | 57 | 111001 | 71 | 39 |
| 26 | 11010 | 32 | 1a | 58 | 111010 | 72 | 3a |
| 27 | 11011 | 33 | 1b | 59 | 111011 | 73 | 3b |
| 28 | 11100 | 34 | 1c | 60 | 111100 | 74 | 3c |
| 29 | 11101 | 35 | 1d | 61 | 111101 | 75 | 3d |
| 30 | 11110 | 36 | 1e | 62 | 111110 | 76 | 3e |
| 31 | 11111 | 37 | 1f | 63 | 111111 | 77 | 3f |

Table B.1: A list of the integers 0–63 in decimal, binary, octal, and hexadecimal.

These bits are divided into 3 parts: sign, exponent and fraction,

$$s\, e_1 e_2 \ldots e_{11}\, m_1 m_2 \ldots m_{52},$$

where $s$, $e_i$, and $m_j$ are binary digits. The bits are converted to a number using the equation by first calculating the exponent $e$ and the mantissa $m$,

$$e = \sum_{i=1}^{11} e_i 2^{11-i}, \tag{B.3}$$

$$m = \sum_{j=1}^{52} m_j 2^{-j}. \tag{B.4}$$

I.e., the exponent is an integer, where $0 \leq e < 2^{11}$, and the mantissa is a rational, where $0 \leq m < 1$. For most combinations of $e$ and $m$ the real number $v$ is calculated as,

$$v = (-1)^s \, (1 + m) \, 2^{e-1023} \tag{B.5}$$

with the exception that

| | $m = 0$ | $m \neq 0$ |
|---|---|---|
| $e = 0$ | $v = (-1)^s\, 0$ (signed zero) | $v = (-1)^s\, m2^{1-1023}$ (subnormals) |
| $e = 2^{11} - 1$ | $v = (-1)^s\, \infty$ | $v = (-1)^s$ NaN (not a number) |

· subnormals
· NaN
· not a number

where $e = 2^{11} - 1 = 11111111111_2 = 2047$. The largest and smallest number that is not infinity is thus

$$e = 2^{11} - 2 = 2046 \tag{B.6}$$

$$m = \sum_{j=1}^{52} 2^{-j} = 1 - 2^{-52} \simeq 1. \tag{B.7}$$

$$v_{\max} = \pm \left(2 - 2^{-52}\right) 2^{1023} \simeq \pm 2^{1024} \simeq \pm 10^{308} \tag{B.8}$$

The density of numbers varies in such a way that when $e - 1023 = 52$, then

$$v = (-1)^s \left(1 + \sum_{j=1}^{52} m_j 2^{-j}\right) 2^{52} \tag{B.9}$$

$$= \pm \left(2^{52} + \sum_{j=1}^{52} m_j 2^{-j} 2^{52}\right) \tag{B.10}$$

$$= \pm \left(2^{52} + \sum_{j=1}^{52} m_j 2^{52-j}\right) \tag{B.11}$$

$$\overset{k=52-j}{=} \pm \left(2^{52} + \sum_{k=51}^{0} m_{52-k} 2^k\right) \tag{B.12}$$

which are all integers in the range $2^{52} \leq |v| < 2^{53}$. When $e - 1023 = 53$, then the same calculation gives

$$v \overset{k=53-j}{=} \pm \left(2^{53} + \sum_{k=52}^{1} m_{53-k} 2^k\right) \tag{B.13}$$

which are every second integer in the range $2^{53} \leq |v| < 2^{54}$, and so on for larger $e$. When $e - 1023 = 51$, then the same calculation gives,

$$v \overset{k=51-j}{=} \pm \left(2^{51} + \sum_{k=50}^{-1} m_{51-k} 2^k\right) \tag{B.14}$$

which gives a distance between numbers of $1/2$ in the range $2^{51} \leq |v| < 2^{52}$, and so on for smaller $e$. Thus we may conclude that the distance between numbers in the interval $2^n \leq |v| < 2^{n+1}$ is $2^{n-52}$, for $-1022 = 1 - 1023 \leq n < 2046 - 1023 = 1023$. For subnormals, the distance between numbers are

$$v = (-1)^s \left(\sum_{j=1}^{52} m_j 2^{-j}\right) 2^{-1022} \tag{B.15}$$

$$= \pm \left(\sum_{j=1}^{52} m_j 2^{-j} 2^{-1022}\right) \tag{B.16}$$

$$= \pm \left(\sum_{j=1}^{52} m_j 2^{-j-1022}\right) \tag{B.17}$$

$$\overset{k=-j-1022}{=} \pm \left(\sum_{j=-1023}^{-1074} m_{-k-1022} 2^k\right) \tag{B.18}$$

which gives a distance between numbers of $2^{-1074} \simeq 10^{-323}$ in the range $0 < |v| < 2^{-1022} \simeq 10^{-308}$.

# C | Commonly used character sets

Letters, digits, symbols and space are the core of how we store data, write programs, and communicate with computers and each other. These symbols are in short called characters, and represents a mapping between numbers, also known as codes, and a pictorial representation of the character. E.g., the ASCII code for the letter 'A' is 65. These mappings are for short called character sets, and due to differences in natural languages and symbols used across the globe, many different character sets are in use. E.g., the English alphabet contains the letters 'a' to 'z', which is shared by many other European languages, but which have other symbols and accents for example, Danish has further the letters 'æ', 'ø', and 'å'. Many non-European languages have completely different symbols, where Chinese character set is probably the most extreme, where some definitions contains 106,230 different characters albeit only 2,600 are included in the official Chinese language test at highest level.

Presently, the most common character set used is Unicode Transformation Format (UTF), whose most popular encoding schemes are 8-bit (UTF-8) and 16-bit (UTF-16). Many other character sets exist, and many of the later builds on the American Standard Code for Information Interchange (ASCII). The ISO-8859 codes were an intermediate set of character sets that are still in use, but which is greatly inferior to UTF. Here we will briefly give an overview of ASCII, ISO-8859-1 (Latin1), and UTF.

## C.1 ASCII

The *American Standard Code for Information Interchange* (*ASCII*) [8], is a 7 bit code tuned for the letters of the english language, numbers, punctuation symbols, control codes and space, see Tables C.1 and C.2. The first 32 codes are reserved for non-printable control characters to control printers and similar devices or to provide meta-information. The meaning of each control characters is not universally agreed upon.

· American Standard Code for Information Interchange
· ASCII

The code order is known as *ASCIIbetical order*, and it is sometimes used to perform arithmetic on codes, e.g., an upper case letter with code $c$ may be converted to lower case by adding 32 to its code. The ASCIIbetical order also has consequence for sorting, i.e., when sorting characters according to their ASCII code, then 'A' comes before 'a', which comes before the symbol '{'.

· ASCIIbetical order

## C.2 ISO/IEC 8859

The ISO/IEC 8859 report `http://www.iso.org/iso/catalogue_detail?csnumber=28245` defines 10 sets of codes specifying up to 191 codes and graphic characters using 8 bits. Set 1 also known as ISO/IEC 8859-1, Latin alphabet No. 1, or *Latin1* covers many European languages and is designed to be compatible with ASCII, such that code for the printable characters in ASCII are the same in ISO 8859-1. In Table C.3 is shown the characters above 7e. Codes 00-1f and 7f-9f are undefined in ISO 8859-1.

· Latin1

| x0+0x | 00 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|---|---|---|---|---|---|---|---|---|
| 00 | NUL | DLE | SP | 0 | @ | P | ‘ | p |
| 01 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 02 | STX | DC2 | " | 2 | B | R | b | r |
| 03 | ETX | DC3 | # | 3 | C | S | c | s |
| 04 | EOT | DC4 | $ | 4 | D | T | d | t |
| 05 | ENQ | NAK | % | 5 | E | U | e | u |
| 06 | ACK | SYN | & | 6 | F | V | f | v |
| 07 | BEL | ETB | ’ | 7 | G | W | g | w |
| 08 | BS | CAN | ( | 8 | H | X | h | x |
| 09 | HT | EM | ) | 9 | I | Y | i | y |
| 0A | LF | SUB | * | : | J | Z | j | z |
| 0B | VT | ESC | + | ; | K | [ | k | { |
| 0C | FF | FS | , | < | L | \ | l | \| |
| 0D | CR | GS | – | = | M | ] | m | } |
| 0E | SO | RS | . | > | N | ^ | n | ~ |
| 0F | SI | US | / | ? | O | _ | o | DEL |

Table C.1: ASCII

## C.3 Unicode

Unicode is a character standard defined by the Unicode Consortium, `http://unicode.org` as the *Unicode Standard*. Unicode allows for 1,114,112 different codes. Each code is called a *code point*, which represents an abstract character. However, not all abstract characters require a unit of several code points to be specified. Code points are divided into 17 planes each with $2^{16} = 65,536$ code points. Planes are further subdivided into named *blocks*. The first plane is called the *Basic Multilingual plane* and it are the first 128 code points is called the *Basic Latin block* and are identical to ASCII, see Table C.1, and code points 128-255 is called the *Latin-1 Supplement block*, and are identical to the upper range of ISO 8859-1, see Table C.3. Each code-point has a number of attributes such as the *Unicode general category*. Presently more than 128,000 code points covering 135 modern and historic writing systems, and obtained at `http://www.unicode.org/Public/UNIDATA/UnicodeData.txt`, which includes the code point, name, and general category.

- Unicode Standard
- code point
- blocks
- Basic Multilingual plane
- Basic Latin block
- Latin-1 Supplement block
- Unicode general category

A Unicode code point is an abstraction from the encoding and the graphical representation of a character. A code point is written as “U+” followed by its hexadecimal number, and for the Basic Multilingual plane 4 digits are used, e.g., the code point with the unique name LATIN CAPITAL LETTER A has the Unicode code point is “U+0041”, and is in this text it is visualized as ’A’. More digits are used for code points of the remaining planes.

The general category is used in grammars to specify valid characters, e.g., in naming identifiers in F#. Some categories and their letters in the first 256 code points are shown in Table C.5.

To store and retrieve code points, they must be encoded and decoded. A common encoding is *UTF-8*, which encodes code points as 1 to 4 bytes, and which is backward-compatible with ASCII and ISO 8859-1. Hence, in all 3 coding systems the character with code 65 represents the character ’A’. Another popular encoding scheme is *UTF-16*, which encodes characters as 2 or 4 bytes, but which is not backward-compatible with ASCII or ISO 8859-1. UTF-16 is used internally in many compiles, interpreters and operating systems.

- UTF-8
- UTF-16

| Code | Description |
|---|---|
| NUL | Null |
| SOH | Start of heading |
| STX | Start of text |
| ETX | End of text |
| EOT | End of transmission |
| ENQ | Enquiry |
| ACK | Acknowledge |
| BEL | Bell |
| BS | Backspace |
| HT | Horizontal tabulation |
| LF | Line feed |
| VT | Vertical tabulation |
| FF | Form feed |
| CR | Carriage return |
| SO | Shift out |
| SI | Shift in |
| DLE | Data link escape |
| DC1 | Device control one |
| DC2 | Device control two |
| DC3 | Device control three |
| DC4 | Device control four |
| NAK | Negative acknowledge |
| SYN | Synchronous idle |
| ETB | End of transmission block |
| CAN | Cancel |
| EM | End of medium |
| SUB | Substitute |
| ESC | Escape |
| FS | File separator |
| GS | Group separator |
| RS | Record separator |
| US | Unit separator |
| SP | Space |
| DEL | Delete |

Table C.2: ASCII symbols.

| x0+0x | 80 | 90 | A0 | B0 | C0 | D0 | E0 | F0 |
|---|---|---|---|---|---|---|---|---|
| 00 | | | NBSP | ° | À | Ð | à | ð |
| 01 | | | ¡ | ± | Á | Ñ | á | ñ |
| 02 | | | ¢ | $^2$ | Â | Ò | â | ò |
| 03 | | | £ | $^3$ | Ã | Ó | ã | ó |
| 04 | | | ¤ | ´ | Ä | Ô | ä | ô |
| 05 | | | ¥ | $\mu$ | Å | Õ | å | õ |
| 06 | | | ¦ | ¶ | Æ | Ö | æ | ö |
| 07 | | | § | · | Ç | × | ç | ÷ |
| 08 | | | ¨ | ¸ | È | Ø | è | ø |
| 09 | | | © | $^1$ | É | Ù | é | ù |
| 0a | | | ª | º | Ê | Ú | ê | ú |
| 0b | | | « | » | Ë | Û | ë | û |
| 0c | | | ¬ | $\frac{1}{4}$ | Ì | Ü | ì | ü |
| 0d | | | SHY | $\frac{1}{2}$ | Í | Ý | í | ý |
| 0e | | | ® | $\frac{3}{4}$ | Î | Þ | î | þ |
| 0f | | | ¯ | ¿ | Ï | ß | ï | ÿ |

Table C.3: ISO-8859-1 (latin1) non-ASCII part. Note that the codes 7f – 9f are undefined.

| Code | Description |
|---|---|
| NBSP | Non-breakable space |
| SHY | Soft hypen |

Table C.4: ISO-8859-1 special symbols.

| General category | Code points | Name |
|---|---|---|
| Lu | U+0041–U+005A, U+00C0–U+00D6, U+00D8–U+00DE | Upper case letters |
| Ll | U+0061–U+007A, U+00B5, U+00DF–U+00F6, U+00F8–U+00FF | Lower case letter |
| Lt | None | Digraphic letter, with first part uppercase |
| Lm | None | Modifier letter |
| Lo | U+00AA, U+00BA | Gender ordinal indicator |
| Nl | None | Letterlike numeric character |
| Pc | U+005F | Low line |
| Mn | None | Nonspacing combining mark |
| Mc | None | Spacing combining mark |
| Cf | U+00AD | Soft Hyphen |

Table C.5: Some general categories for the first 256 code points.

# Bibliography


[1] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345—-363, 1936.

[2] Ole-Johan Dahl and Kristen Nygaard. SIMULA a language for programming and description of discrete event systems. introduction and user's manual. Technical report, Norwegian Computing Center, 1967.

[3] European Computer Manufacturers Association (ECMA). Standard ecma-335, common language infrastructure (cli). `http://www.ecma-international.org/publications/standards/Ecma-335.htm`.

[4] International Organization for Standardization. Iso/iec 23271:2012, common language infrastructure (cli). `https://www.iso.org/standard/58046.html`.

[5] Object Management Group. Uml version 2.0. `http://www.omg.org/spec/UML/2.0/`.

[6] Programming Research Group. Specifications for the ibm mathematical formula translating system, fortran. Technical report, Applied Science Division, International Business Machines Corporation, 1954.

[7] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM*, 3(4):184–195, 1960.

[8] X3: ASA Sectional Committee on Computers and Information Processing. American standard code for information interchange. Technical Report ASA X3.4-1963, American Standards Association (ASA), 1963. `http://worldpowersystems.com/projects/codes/X3.4-1963/`.

[9] George Pólya. *How to solve it*. Princeton University Press, 1945.

[10] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1936.

# Index