By: Ehsan Kourkchi (Sep., 21, 2013)

Report …

1. Number Conversions

Decimal to Binary Conversions

112

- 112 = 2 * 56 + 0
- 56 = 2 * 28 + 0
- 28 = 2 * 14 + 0
- 17 = 2 * 7 + 0
- 7 = 2 * 3 + 1
- 3 = 2 * 1 + 1
- 1 = 2 * 0 + 1

$(112)_{10} = (111,0000)_2$

101

- 101 = 2 * 50 + 1
- 50 = 2 * 25 + 0
- 25 = 2 * 12 + 1
- 12 = 2 * 6 + 0
- 6 = 2 * 3 + 0
- 3 = 2 * 1 + 1
- 1 = 2 * 0 + 1

$(101)_{10} = (110,0101)_2$

128

- 128 = 2 * 64 + 0
- 64 = 2 * 32 + 0
- 32 = 2 * 16 + 0
- 16 = 2 * 8 + 0
- 8 = 2 * 4 + 0
- 4 = 2 * 2 + 0
- 2 = 2 * 1 + 0
- 1 = 2 * 0 + 1

$(128)_{10} = (1000,0000)_2$

By: Ehsan Kourkchi (Sep., 21, 2013)

Decimal to Hexadecimal Conversions

$(112)_{10} = (111,0000)_2$
$(111)_2 = 7 = (7)_{16}$ …. $(0000)_2 = 0 = (0)_{16}$
$(112)_{10} = (1110,000)_2 = (70)_{16}$

$(101)_{10} = (110,0101)_2$
$110)_2 = 6 = (6)_{16}$ …. $(0101)_2 = 5 = (5)_{16}$
$(101)_{10} = (110,0101)_2 = (65)_{16}$

$(128)_{10} = (1000,0000)_2$
$(1000)_2 = 8 = (8)_{16}$ …. $(0000)_2 = 0 = (0)_{16}$
$(128)_{10} = (1000,0000)_2 = (80)_{16}$

Binary to Decimal Conversions

(1001)2 = 1 * 8 + 0 * 4 + 0 * 2 + 1 = 8 + 1 = 9

(11001)2 = 1 * 16 + 1 * 8 + 0 * 4 + 0 * 2 + 1 = 16 + 8 + 1 = 25

(1111)2 = 1 * 8 + 1 * 4 + 1 * 2 + 1 = 8 + 4 + 2 + 1 = 15

By: Ehsan Kourkchi (Sep., 21, 2013)

2. The use and benefits of command-line arguments

Once a program is compiled, user don't usually have access to the source. Command line inputting and outputting provides the user a trivial way of communication with the program.

The program can be then implemented by other scripts (e.g. bash, csh) in a series of command lines. This allows the user to take the advantage of the command line features of a set of programs to perform the desired tasks. The command line output behavior of a program helps the user to either track the errors or inject the output to other programs. This flexibility allows the other designers to execute different programs in a series of commands and design more complicated tools. Therefore, software development would be easier this way. In command line running mode, the output and input of the programs can be piped together and create some complex features.

Command line arguments let user to run a program under several different modes. Even if the user is not familiar how to run a program, he/she can often use '--help' or '-h' standard flags to find more about the capabilities of the program. Of course, these standard arguments should have been provided by the designer. Using different argument sets, a program can serves users in different ways. For example, a program can read the inputs from one by one, ore through an input file, or even user may want a rapid change of an input variable to test how the results change. The command line arguments can give the user a chance to play with the different ways that a program can be executed and to choose the best mode for his/her purpose. Thus, the command line arguments speed up the usage of the program. Moreover, it lets user run their programs remotely in other computers where the GUI (Graphical User Interface) is not critically demanded.

The only disadvantage of the command line programing is that it may be difficult and complicated to provide a multipurpose design. In addition, in command line programing, a beautiful GUI does not exist like what normally happens in Windows.