**1. Installing MIT-Scheme**

**1. Why Scheme?**

By Using Scheme:

* You can write beautiful programs.
* You can enjoy programming.

These are the only reasons to learn Scheme. You will find some difficulties if you write practical applications using Scheme.

However, as excellent hackers favor Scheme, it is worth learning the language. In fact, [Structure and Interpretation of Computer Programs (SICP)](http://mitpress.mit.edu/sicp/) — one of the best textbook on computer science — uses Scheme to describe programs. [GNU](http://www.gnu.org/) also uses Scheme (named [guile](http://www.gnu.org/software/guile/guile.html)) for a common script language for its applications. The guile corresponds to a macro for MS-Word or Excell. It is used to manupulate applications with a simple script.

Even Common Lisp is better for building practical applications, I recommend to learn Scheme first because of its

1. compact language design and
2. simple syntax.

It is said that [Scheme makes you better programmer](http://www.catb.org/~esr/faqs/hacker-howto.html#skills1). Even you seldom use Scheme for your business, the good sense provided by learning Scheme will guide you even if you use other programming languages.

Scheme tutorials in the web (so many as Scheme is popular language) are somehow difficult for beginners. This tutorial, on the other hand, is written for beginner programmers and little knowledge on programming is required.

**2. Expected readers**

This tutorial expects PC users who have little programming experience such as:

* Students who cannot follow his course because the professor uses Scheme in the lecture.
* People who try to learn programming

The syntax of Scheme is extremely simple and can be explained in a simple manner. However, such explanation is quite difficult for beginners. In this tutorial, I will try to explain step by step.

Scheme codes consist of only words, parentheses and spaces, which may embarrass you at the beginning. If you use proper editor, however, it shows the pairing bracket automatically and indents code for you. As a result, you do not have to worry about pairing of parentheses and you can read codes by indent. If the indent looks strange, you can find the wrong pairing using the editor.

**3. Installing MIT-Scheme**

This tutorial expects Windows user. I have no experience of Macintosh so I can tell nothing about it. If you are a Unix (or Linux) user, ask the administrator to install it (if you cannot by yourself). The way of using Scheme is not depend on the Operation system. Only the installation matters.

There are several specification on Scheme programming language and the latest version is [Revised5 Report on the Algorithmic Language Scheme (R5RS)](http://www.schemers.org/Documents/Standards/R5RS/HTML/r5rs.html).

Most of the implementations are (fully or partially) based on R5RS. If you use one that partially satisfies R5RS, you should be careful about it. There are several free Scheme implementations working on Windows such as: [ChezScheme](http://www.scheme.com/), [MzScheme](http://www.plt-scheme.org/software/mzscheme/), [DrScheme](http://www.drscheme.org/), [SCM](http://swissnet.ai.mit.edu/~jaffer/SCM.html). In this tutorial, I use [MIT/GNU Scheme](http://www.gnu.org/software/mit-scheme/) because it is easy to install and it has high performance. The interpreter is quite fast and in addition MIT-Scheme can compile your program to a native code. The problem of MIT-Scheme is that it does not fully satisfy the R5RS. I will mention about it later in detail.

Actually, only MIT-Scheme and DrScheme are equipped with installer. Some recommend DrScheme, but it is slow. If you have enough knowledge to install programs by hand, I also recommend [Petite Chez Scheme](http://www.scheme.com/petitechezscheme.html). It is a nice interpreter working on command line (DOS window).

[Scheme Implementations](http://www.aracnet.com/~briand/scheme_eval.html) compares several Scheme implementations. When you get used to Scheme, it will be a nice idea to try several implementations. You need a Linux machine because most of Scheme implementations work only on Unix and Linux.

**3.1. How to install MIT-Scheme on Windows**

The MIT-Scheme can be installed by just downloading and executing the installer.

1. Go to [the homepage of MIT/GNU Scheme](http://www.gnu.org/software/mit-scheme/) and download the Windows binary, mit-scheme-N.N.N-ix86-win32.exe.
2. Double click the downloaded installer. The installer asks something and leave everything as default.
3. After installation, four shortcuts are created, named **Scheme**, **Compiler**, **Edwin**, and **Documentation**. **Scheme**, **Compiler**, and **Edwin** are shortcuts to the same program with different options. By using the **Compiler**, you can compile your program into a native code, which makes run-time shorter. On the other hand, **Compiler** consumes more memory.   
   **Edwin** is an Emacs like editor for editing Scheme programs. You can use this or your favorite editor.
4. You can customize the MIT-Scheme by editing the configuration file **scheme.ini**.  
   Create scheme.ini in a directory defined by the environmental variable **HOMEPATH**. You can check the value of HOMEPATH by giving
5. >set HOMEPATH

from a console (DOS window). In the case of WinXP, HOMEPATH is predefined as   
\Document and Setting\*username*

Following code shows an example of scheme.ini

(cd "C:\\doc\\scheme")

(define call/cc call-with-current-continuation)

The first line means changing directory to C:\doc\scheme. By this code, the MIT-Scheme move to this directory and you don't have to give full path name to load Scheme programs. The second line defines abbreviation of call-with-current-continuation.

**4. Summary**

As the installation is quite easy (except editing scheme.ini), you should get it without any problem.

The next chapter is about how to talk with the front end of the MIT-Scheme.