人工知能プログラミングの実習環境について

1 LISP 処理系 GCL

プログラムや命令を解釈して実行する仕組みを処理系と呼びます。LISP の処理系にはいろいろな実装がありますが、皆さんには GCL を使っていただきます。GCL は GNU Common Lisp の略で、京大で開発された Kyoto Common Lisp (KCL) を基に GNU というグループが開発しました。Common Lisp というのは、LISP の規格 および それを満たす処理系のことです。この規格は、LISP が注目されていた時代に必要と思われる機能をどんどん盛り込んだ非常に大きなものです (この規格を解説した厚さ 4cm 程度の本があります)。ただ、この講義で取り上げるのは LISP の最も基本的な部分であり、Common Lisp の豊富な機能のほんの一部しか使いません。

2 GCL の起動と終了

ITC の UNIX 系計算機では、コマンドラインから `gcl` と入力すると、GNU Common Lisp が起動する。日本語を使いたい時もあるので、通常は X ウィンドウシステムを上げてから使う。

```
ua123456[y12a000]% startx
```

もし、kterm が上がっていなければ、login ウィンドウから kterm を上げる。

```
ua123456[y12a000]% kterm &
```

kterm ウィンドウから gcl を立ち上げる。

```
ua123456[y12a000]% gcl
GCL (GNU Common Lisp)  2.6.8 ANSI Mar 19 2014 13:45:42
Source License: ..............
Binary License: ...................
Modifications of the banner must retain notice of a compatible license
Dedicated to the memory of W. Schelter

Use (help) to get some basic information on how to use GCL.
Temporary directory for compiler files set to /tmp/

>
```

ここで、最後の不等号は LISP のプロンプトである (LISP はインタプリタなので、命令の入力を逐次受け付けることができる)。

LISP の終了は 関数 `bye` で行う。

```
>(bye)
ua123456[y12a000]%
```

3　方法１：エディタと GCL を別画面であげる

一つの画面でエディタ（emacs, vi, vim 等）を上げ、プログラムを作成（修正）して保存する。lisp プログラムの拡張子は l（エル）や lsp とすることが多い。

別画面で GCL を立ち上げておいて、作っておいた関数や変数を次のようにファイルとして読み込む。

```
>(load "test.lsp")
```

4　方法２：emacs 上での GCL の利用

(注) 便利な生活をするためには、最初に少し苦労しなくてはいけない。

LISP 処理系を emacs 上で利用することの利点は、括弧の対応づけが分かり易い (emacs には閉括弧を入力したときに、それと対応する開括弧にカーソルがちょっとの間移動して、対応を教えてくれる機能がある)、似たような入力を繰り返すときにはエディタの機能 (カット・ペーストなど) を使うことができる、出力が簡単にファイルにできるといった点が挙げられる。

emacs 上で GCL を使うために、emacs の設定を変更する必要がある。自分のホームディレクトリの直下に `.emacs` というファイルがある (このような '.' で始まる名前を持つファイルは普通に `ls` とやっても表示されないが、emacs で `C-x C-f` を使って通常のファイルと同様に読み込むことができる)。このファイルは emacs を起動したときに読み込まれるもので、このファイルの末尾に以下を加える。

```
(setq inferior-lisp-program "/usr/local/bin/gcl")
(setq inferior-lisp-mode-hook
      '(lambda ()
         (set-buffer-file-coding-system 'utf-8)
         (set-buffer-process-coding-system 'utf-8 'utf-8)
         ))
```

このように書き加えることで、emacs はその上で LISP 処理系を起動したときに何をすべきかを知る。日本語の表示はもちろんのこと、(あまり使わないが) 日本語の関数名も付けられるようになる。言うまでもないが、書き加えたら保存 (`C-x C-s`) する。設定ファイルに書き加えたら、一度 emacs を終了してから再度立ち上げる (設定ファイルを読み直す必要があるから)。emacs 上で LISP 処理系を起動するためには `M-x` に続いて `run-lisp` と入力し、Enter キーを押す。以下に、重宝な emacs のコマンドをいくつか紹介する。

`C-x 2`	emacs の画面を上下 2 段に分ける (このそれぞれを (emacs の) ウィンドウと呼ぶ)。たとえば上のウィンドウでプログラムを編集しながら、下のウィンドウで処理系を動かすなどの使い方をする。
`C-x o`	(emacs の) ウィンドウの間をカーソルが移動する。これを使えばマウスを使わずに (キーボードから手を離さずに) ほとんどの作業をすることができる。
`M-C-x`	(プログラムファイル中の)LISP の定義式の中にカーソルがある状態でこれを行うと、LISP の処理系にその定義が送り込まれる。これを行う前に `M-x run-lisp` で処理系を起動しておく必要がある。

emacs 上で LISP を起動した場合は、emacs を終了する前に、LISP 処理系を関数 (`bye`) で終了しておく。

5　便利な機能

5.1　日本語ファイルを utf-8 に変換する

PC 環境でプログラムを作成した後などに、日本語コードを utf-8 にしたい時は、UNIX のプロンプトから

```
% nkf -w8 sourcefile.lsp > targetfile.lsp
```

のようにする。

5.2 GCL のデバッグ機能について

エラーが起きた時は自動的にデバッガーに入る。プロンプトは>>で、そこでいろいろな機能が起動できる。代表的なものにバックトレースがあり、

```
>>:bt
```

とする.

```
>>:help
```

とすると、デバッグモードのコマンド一覧が出てくる。デバッグモードから抜けるには、

```
>>:q
```

とする。デバッグモードでは、変数の値が表示できる。
特定の関数について、その入力値および出力値を表示するには、

```
>(trace foo)
```

とする。関数のトレース機能をはずすには、

```
>(untrace foo)
```

とする。引数なしに

```
>(untrace)
```

とだけするとトレース中のすべての関数のトレース機能を off にできる。