25. Environment

25.1 The External Environment

25.1.1 Top level loop

The top level loop is the Common Lisp mechanism by which the user normally interacts with the Common Lisp system. This loop is sometimes referred to as the *Lisp read-eval-print loop* because it typically consists of an endless loop that reads an expression, evaluates it and prints the results.

The top level loop is not completely specified; thus the user interface is *implementation-defined*. The top level loop prints all values resulting from the evaluation of a *form*. The next figure lists variables that are maintained by the *Lisp read-eval-print loop*.

```
*    +    /    -
**   ++   //
***  +++  ///
```

Figure 25-1. Variables maintained by the Read-Eval-Print Loop

25.1.2 Debugging Utilities

The next figure shows *defined names* relating to debugging.

```
*debugger-hook*  documentation    step
apropos          dribble          time
apropos-list     ed               trace
break            inspect          untrace
describe         invoke-debugger
```

Figure 25-2. Defined names relating to debugging

25.1.3 Environment Inquiry

Environment inquiry *defined names* provide information about the hardware and software configuration on which a Common Lisp program is being executed.

The next figure shows *defined names* relating to environment inquiry.

```
*features*                  machine-instance  short-site-name
lisp-implementation-type    machine-type      software-type
lisp-implementation-version machine-version   software-version
long-site-name              room
```

Figure 25-3. Defined names relating to environment inquiry.

25.1.4 Time

Time is represented in four different ways in Common Lisp: *decoded time*, *universal time*, *internal time*, and seconds. *Decoded time* and *universal time* are used primarily to represent calendar time, and are precise only to one second. *Internal time* is used primarily to represent measurements of computer time (such as run time) and is precise to some *implementation-dependent* fraction of a second called an *internal time unit*, as specified by **internal-time-units-per-second**. An *internal time* can be used for either *absolute* and *relative time* measurements. Both a *universal time* and a *decoded time* can be used only for *absolute time* measurements. In the case of one function, **sleep**, time intervals are represented as a non-negative *real* number of seconds.

The next figure shows *defined names* relating to *time*.

```
decode-universal-time   get-internal-run-time
encode-universal-time   get-universal-time
get-decoded-time        internal-time-units-per-second
get-internal-real-time  sleep
```

Figure 25-4. Defined names involving Time.

25.1.4.1 Decoded Time

A *decoded time* is an ordered series of nine values that, taken together, represent a point in calendar time (ignoring *leap seconds*):

Second
An *integer* between 0 and 59, inclusive.

Minute
An *integer* between 0 and 59, inclusive.

Hour
An *integer* between 0 and 23, inclusive.

Date
An *integer* between 1 and 31, inclusive (the upper limit actually depends on the month and year, of course).

Month
An *integer* between 1 and 12, inclusive; 1 means January, 2 means February, and so on; 12 means December.

Year
An *integer* indicating the year A.D. However, if this *integer* is between 0 and 99, the "obvious" year is used; more precisely, that year is assumed that is equal to the *integer* modulo 100 and within fifty years of the current year (inclusive backwards and exclusive forwards). Thus, in the year 1978, year 28 is 1928 but year 27 is 2027. (Functions that return time in this format always return a full year number.)

Day of week
An *integer* between 0 and 6, inclusive; 0 means Monday, 1 means Tuesday, and so on; 6 means Sunday.

Daylight saving time flag
A *generalized boolean* that, if *true*, indicates that daylight saving time is in effect.

Time zone
A *time zone*.

The next figure shows *defined names* relating to *decoded time*.

```
decode-universal-time  get-decoded-time
```

Figure 25-5. Defined names involving time in Decoded Time.

25.1.4.2 Universal Time

Universal time is an *absolute time* represented as a single non-negative *integer*---the number of seconds since midnight, January 1, 1900 GMT (ignoring *leap seconds*). Thus the time 1 is 00:00:01 (that is, 12:00:01 a.m.) on January 1, 1900 GMT. Similarly, the time 2398291201 corresponds to time 00:00:01 on January 1, 1976 GMT. Recall that the year 1900 was not a leap year; for the purposes of Common Lisp, a year is a leap year if and only if its number is divisible by 4, except that years divisible by 100 are not leap years, except that years divisible by 400 are leap years. Therefore the year 2000 will be a leap year. Because *universal time* must be a non-negative *integer*, times before the base time of midnight, January 1, 1900 GMT cannot be processed by Common Lisp.

```
decode-universal-time  get-universal-time
encode-universal-time
```

Figure 25-6. Defined names involving time in Universal Time.

25.1.4.3 Internal Time

Internal time represents time as a single *integer*, in terms of an *implementation-dependent* unit called an *internal time unit*. Relative time is measured as a number of these units. Absolute time is relative to an arbitrary time base.

The next figure shows *defined names* related to *internal time*.

```
get-internal-real-time  internal-time-units-per-second
get-internal-run-time
```

Figure 25-7. Defined names involving time in Internal Time.

25.1.4.4 Seconds

One function, **sleep**, takes its argument as a non-negative *real* number of seconds. Informally, it may be useful to think of this as a *relative universal time*, but it differs in one important way: *universal times* are always non-negative *integers*, whereas the argument to **sleep** can be any kind of non-negative *real*, in order to allow for the possibility of fractional seconds.

```
sleep
```

Figure 25-8. Defined names involving time in Seconds.