4. More on Navigating the Lisp Machine

The last chapter discussed aspects of programming with the Lisp language. This one is about some aspects of using the Lisp Machine which are more or less independent of programming on it, *i.e.*, what you might call the operating system of the Lisp Machine. Some parts of this chapter having to do with sending messages to windows and process objects may be confusing unless you know something about flavors. See chapter 5, especially the section starting at page 117.

4.1 The Scheduler and Processes

A *process* is a single computational sequence within a computer. The Lisp Machine supports multiple processes running "simultaneously," *i.e.*, sharing the processor like a miniature time-sharing system. Each process behaves like it has its own simulated processor: it has its own "program counter," its own function-call history (*stack*), and its own special-variable bindings.

Switching the processor back and forth among the different
processes can be explicitly controlled by the Lisp Machine programmer (read the documentation on *Stack Groups*), but almost never is. A special module called the *scheduler* generally handles this responsibility. Every 1/60th second the scheduler wakes up and decides whether the current process should be allowed to continue running, and if not, which other process should get a chance.

If the current process has been running continuously for less than a second, and wishes to continue, it is allowed to. (Note that a full second is a long time for this sort of thing, compared to other timesharing arrangements.) Or if it’s been running for a second but no other process wishes to run, it is still allowed to continue. But if it’s been monopolizing the machine for more than a second, and one or more other processes want to run, it’s forced to take a rest while the scheduler gives the others a chance. The process chosen by the scheduler is now treated as the previous current process was: it will be allowed to run until some other process(es) wish to run and the current process either volunteers to give the others a chance, or passes the one second mark.¹

The way a process “volunteers to give the others a chance,” or, in less emotionally-laden terms, informs the scheduler that it doesn’t need to run, is with the function *process-wait*. The function which calls *process-wait* specifies a condition the process is waiting for. When the condition becomes true, the process is ready to run. When the scheduler decides to resume the process, the call to *process-wait* returns and the computation continues from there. The first argument to *process-wait*

¹The “one second” referred to is actually a shorthand way of saying “the process’ quantum.” The default runtime quantum is one second, but may be modified for any process. See the section “Process Attribute Messages” in *Internals, Processes, and Storage Management*. 