A New Lisp Machine

Examination of Necessity

H. W. Egdorf

This essay examines the reasons and plans for development of a new Lisp Machine operating environment. The essay is structured in three sections. The first section gives a summary and the anticipated goals of the project. This is followed by a section with the rationale for the directions presented and an expansion on the goals. Finally a technical direction for the anticipated work is described. The reasons and rationale for the development of this system are not exhaustive. Rather, they are documentation of author's motivations for pursuit of this work. The plan for this work is presented as future potential. In reality, some of this work has begun and this essay documents the initial development direction underway.

Summary and Goals

The goal of this effort is to produce a modern base architecture for a Lisp Machine system using commodity hardware. Commodity hardware in this case means readily available Intel-based PC systems.

The approach is to use the Linux kernel running on commodity hardware as a supervisor. This will provide access to the broadest possible range of hardware with a known and stable set of interfaces and performance characteristics.

The Unix system has always had its user-space look and feel defined by the Init process (typically in the program file /etc/init) which spawns and defines the configuration of the normal expected Unix environment. The development described here proposes to replace this set of user-level processes with a Lisp system. This Lisp-based environment will then be used as a new process to configure and define an environment similar to the Lisp Machine's in the user space provided by the Linux kernel.

The authors will then use this new Lisp-based environment to experiment with, extend, and develop a new architecture for software independent of the two current dominant architectures; Windows and Unix.

Rationale

Many reasons exist for the pursuit of this development activity. Listed here are those few major reasons driving the current author. Others who wish to join this development activity will almost certainly bring their own reasons to the project. That, in itself, is an important part of such an effort.

Scratch the Itch

It has been said that all good development projects exist to “scratch an itch” on the part of the developer. This project is no different. The primary reason that this project began was the belief on the part of the author that the project would be fun. The aesthetic aspect of a technical activity provides a stronger motivation than many other more common rewards.

The Lisp Machine environment was aesthetically pleasing. The author believes that an attractive environment is a worthy goal beyond technical details of a system implementation. The production of a new Lisp-based environment provides a platform that can be used to explore aspects of the aesthetics of a computer environment beyond those easily available in existing platforms.

Expand Cultural Diversity

Much has been published recently about the danger of a mono-culture in the field of software as it relates to system security and susceptibility to computer viruses. The author believes that there exists a broader danger to the field of computer science where too few architectures exist to encourage broadly innovative experimentation.

Two major architectures exist currently; Microsoft Windows and Unix. While these two cultures are somewhat different, they both provide similar file system, network, and process architectures. The Lisp Machine architecture provides a sufficiently different architecture to encourage broader experimentation.

There would be less need for an additional architecture if existing diversity were expanding. However, the reverse is true. Proprietary environments such as VMS are disappearing. Recently the Macintosh OS has moved to a Unix base. The base of existing architectures is shrinking.

The author does not wish to imply that it is impossible to experiment with new software capability within the existing architectures. The recent move of the Macintosh environment to a Unix base confirms this. The key belief is that there is a synergy of having an integrated new environment that will provide a deeper and richer impetus to this architectural experimentation than is possible within an existing environment provided by Windows or Unix. The Lisp Machine architecture provided such a synergy. The author believes that it is important to move beyond currently common and familiar environments in order to stimulate maximum innovation.

Preserve Knowledge

The Lisp Machines are no longer in mainstream production. While the Lisp language has a fairly rich set of both commercial and open implementations, the overall environment is in danger of being lost.

Features of an environment include more than just the language used. For example a strength of the Lisp Machine architecture

that is less pronounced in existing architectures is the integration between the base system and the user software. The knowledge within a community of the possible benefits of an architectural feature such as this is best maintained by an active example rather than just historical records.
Having an environment where the knowledge and experience from use of the Lisp Machines can be preserved and extended is an important goal. Too many pieces of our young field have already been lost. The author believes that an environment like that of the Lisp Machine provides many examples of architectural features that are best preserved by active use rather than historical documentation and that these features are too important to be relegated to a museum.

Verify the Possibility

There seems to be an opinion that totally replacing the Unix environment is too hard, or too much work. It is true that a great deal of the environment seen by a Unix user is the result of user-level processes outside the kernel. It is, indeed, a lot of work to initialize network capability, connect file systems, perform the multitude of tasks required to run a modern computing environment. A goal of this project is to determine the truth of this opinion regarding the difficulty of the work. How hard is it to produce a new environment in this way?
The answer may have implications beyond just this environment. Other systems may benefit from this experiment. For example, it has been mentioned that a Java virtual machine based environment might be built in the way suggested here.

Directions

CMU Common Lisp as the Linux Init Process

The first step is to replace the normal Unix Init process /etc/init with CMU Common Lisp (CMUCL). CMUCL is a mature system that is freely available for the purpose.
Once Lisp is running as the initial process, a great deal of work remains to be done in order to bring the system to a full usable state. File systems must be checked and mounted, the network must be initialized and configured, and various system processes must be started. This is typically performed on a Unix system by a series of shell scripts that bring the system to a desired run level. Initially, Lisp versions of these procedures will be written, or the existing scripts may be used as a stop-gap procedure. The author believes that an area of interesting experimentation will be to re-cast these processes in a style more fitting to the Lisp system.
As one example, the process of mounting file systems proceeds through a series of steps where the root file system is checked and mounted, then local disks are checked and mounted, and finally network file systems are mounted and auto-mount processes are run. What might it mean to the system architecture when the scripts can be Lisp programs? Will an architecture built on dynamic instances of CLOS objects provide any interesting and perhaps unexpected characteristics over /etc/fstab and /proc/mounts? Until the ability to experiment is readily available, the question cannot be answered by experiment. The author believes that interesting structures may develop and grow that are different from the existing facilities provided within existing Unix systems.

System Support and Configuration

With a running environment in place, experimentation can proceed to the support and configuration of the system. It is important that the system be self-supporting. This means that the initial environment should be usable as a development environment with an editor and sufficient compiler support to re-build the Lisp system and environment. It will be necessary for the system to retain the ability to re-build the underlying Linux kernel. A desirable feature of an environment is that it is self-hosting. This means that the structure must be in place to run Lisp and C compilers and the associated support tools.
As with system start-up, the initial structure of these processes will likely mimic the existing Unix shell process to a large degree. But the existence of the Lisp structure should lead to experimentation in new modes of interaction beyond the Unix or Windows model.

User Interfaces

Initial access to the running system with the normal Linux console terminal emulator may be sufficient for many tasks. The Lisp Machine provided a powerful window-based user interface different than those provided by the current architectures. Versions of CLIM-based interfaces exist that can allow the implementation of a native interface to the Lisp-based environment. This interface may be based on either an underlying X Windows base, or it can be made to run on a raw frame buffer. Both of these approaches have potential for interesting experimentation and the author believes that both will become available.
The system interface, when based on the currently available X Windows , will allow existing Unix-based windowed applications to run. But in addition, an interface built on CLIM providing the strengths of the Lisp Machine Common Windowing environment will provide a fertile area of experimentation that is currently more difficult in the existing Unix environment.

System Architecture

In the long term, a new environment for a system architecture is anticipated to lead to new ways of approaching software engineering problems. The environment must be broader than just a new language.
This is not to imply that much good work cannot be done now with these environments in a Windows or Unix environment.

Recall the key belief listed above under Cultural Diversity. Having the existing environment available produces a multitude of little accommodations and compromises in an architecture like the windowing environment of this example. Performing this work in a different environment (the Lisp Machine environment proposed) produces its own set of accommodations and compromises, but these are new accommodations and compromises. This can make a large difference.

In order to make the best use of these possibilities in the future, a structure must be planned that allows for growth and optimization. The Linux base for a Lisp Machine architecture provides a good growth path for these future directions.

The primary anticipated optimization is to use the Linux module and memory system to place the Lisp runtime system into common memory that is a part of each new process. In other words, Lisp is simply always available to every process. Even when running classic Unix programs, Lisp facilities are available for extension, scripting, and other control. It is too early to predict the possibilities of this, but it is an enticing prospect to play with software in such an architecture.

Conclusion

Examining the opportunities and possibilities in the preceding discussion leads to the ultimate reason for pursuing this path. The author anticipates that this work will be an exciting, stimulating great piece of fun!

It is interesting to play with Lisp as just another language under Windows or Unix. However, Lisp has a long history of providing an innovative environment for system experimentation. This project intends to provide a platform suitable for the continued use and enjoyment of Lisp in its historic role.