計算機アーキテクチャ　99－4
（1993．3．11）

ユーザレベルのプロセス管理はオペレーティングシステムを越えられるか？

白川洋充

近畿大学　理工学部　経営工学科
〒577　東大阪市小若江3－4－1
E-mail:shirakawa@cs.ritsumei.ac.jp

あらまし　ユーザレベルでプロセス管理を行った場合とカーネルレベルでプロセス管理を行った場合を比較すると，前者の性能は後者のそれに比較して非常に良好である．しかしながら，ユーザレベルでのプロセス管理は安全とは言えない．前者を取るか後者を取るかユーザにとっては重要な問題である．ユーザレベルでプロセス管理を意図する研究者に多大な影響を与えた研究，示唆に富んだヒントを与えた研究について紹介する．また，多くのプログラミング言語はコンカレンシイをサポートしている．ここでは代表的なものについて，問題点と特徴のサーベイを行う．

和文キーワード　オペレーティングシステム　プロセス管理　コンカレンシイ　リアルタイムシステム

User-Level Management of Concurrency or Kernel-Level Management of Concurrency

Hiromitsu Shirakawa

Department of Industrial Engineering
Faculty of Science and Engineering
Kinki University
Kowakae 3-4-1, Higashi-Osaka, 577, Japan

Abstract　In this paper we investigate the relation of user-level management of concurrency and kernel-level management of concurrency. In general the former is implemented by runtime library routines. Concurrency can be managed at user's will without intervention of the kernel. Thus it can attain excellent performance compared with that of kernel-supported concurrency. Disadvantage of user-level management of concurrency is that it is unsafe. On the contrary kernel-level management of concurrency is worse in performance, but it is safe and might be robust. Does user-level management of concurrency supersede kernel-level management of concurrency? This is the main theme of this paper. Many programming languages support concurrency. We survey the issues and features of the representative languages.

英文 key words　Operating System, Process Management, Concurrency, Real Time System

はじめに

ユーザレベルでプロセス管理を意図する研究者に多大な影響を与えた研究，示唆に富んだヒントを与えた研究について紹介する．したがって，本稿で紹介する研究はあくまでユーザレベルでプロセス管理を行うことを考えている研究者側から見たサーベイでありオペレーティングシステムを専門としている研究者側から見たサーベイでない．

ユーザがオペレーティングシステムを構築できるか？

プロセス管理を行うことはリアルタイム処理に携わる技術者や並行プログラミングのより効果的な実現を行おうとするシリアスな研究者にとって永年に亙る重要な課題であった．マイクロカーネルがユーザレベルのプロセス管理を始めるようになった８０年代後半までは商用のオペレーティングシステムが提供しているプロセス管理の機能は満足できるものでなかったというのが実情である．このため，ユーザレベルでプロセス管理を行うことはオペレーティングシステムを自前で構築した方がよいのではないかと多くの研究者が考えていた．これに弾みをかけたのが小型計算機の出現である．最初の試みは'７０年代初頭に遡ることができる．文献[1]はオペレーティングシステムを構築しようとする研究者に非常に影響を与えた．筆者の独断ではUNIXはMULTICSよりむしろこの論文から影響を受けたのではないかと考えている．なぜなら，この論文はプログラミング言語ベースでオペレーティングシステムを記述しなければならないことを初めて提案したからである．実際この論文のオペレーティングシステムはBCPLで記述されている[2],[3]．また，I/O STREAMという概念をすでに提案している．この論文の著者の一人であるC. Strachey は夭折したが，C. Straceyらの考えた「継続 (Continuations)」という概念は２０年たって再びオペレーティングシステムの構築に影響を与えるのである．これは，後述するように，ユーザレベルでプロセス管理を効果的に行おうとする場合の重要な実現方法になっている．

プログラミング言語レベルでプロセス管理を行う．

Ｍｅｓａ．

小型計算機でオペレーティングシステムを実際に構築するためにコンカレント処理が可能な高級言語を開発することが考えられた．７０年後半にいくつかのコンカレントプログラミング言語が開発された．その中で代表的なものはN. WirthのModula-2とXerox PARCで開発されたMesa[4]がある．Mesaで実際に記述されたオペレーティングシステムにはPILOTがある．Mesaはプロセスとモニタをサポートした言語である．文献[5]はMesaのプロセスとモニタを使って小型計算機のオペレーティングシステムを記述した場合の問題点について記述している．PILOTの実現におけるこの教訓は現在なお通用するものである．さて，文献[6]はオープンなオペレーティングシステムという概念を提案した．ここでいうオープンとは，ユーザとオペレーティングシステムの間に境界をもうけることはしないこと．さらに，境界をなくしてもシステム全体が頑健(Robustness)になるような技術を開発することを意味している．この論文も小型計算機でオペレーティングシステムを構築する人を勇気づけた．また，オペレーティングシステムを記述するプログラミング言語はランタイムで安全(Safety)でなければならないことを示唆している．なお，Modula-2は欧州，特に英国においてリアルタイムオペレーティングシステムの記述に積極的に使用されている．

Ｍｏｄｕｌａ－３とＯｂｅｌｏｎ．

さて，Mesaの実現を行った研究者がDEC SRC (Digital Systems Research Center)に移ったことと，この研究所は欧州から来た人が多いこともあり，Modula-2とMesaの融合がなされた．この結果，'８４年にModula-2+，'８８年，'８９年にＭｏｄｕｌａ-３の開発を行った[7],[8],[9],[10]．Modula-3の用途は現在模索中であるが，Modula-2+はTopazというオペレーティングシステムとランタイムシステムの構築に貢献した．Modula-2でサポートされているモニタの概念はこの言語でも引き継がれた．ただし，この言語では同期プリミティブを実現するのに用いられている．

欧州のプログラム言語のコミュニティでは Modula-2の後継として Obelon [11],[12]を'８７年に発表した．この言語がユーザレベルでプロセス管理を行おうとしている人にどのようなメリットがあるかは現時点ではコメントできない．

同期プリミティブ．

さて，DEC SRC の貢献はユーザが誤りの少ないプロセス管理のプログラムを書けるようにしたことである．この理由は，同期プリミティブが砂糖の甘味の付いたシンタックス (Syntax sugar) であることによる．また，後述するC ThreadsやSLM Threadsの開発に多大の影響を与えた．

モニタの概念に対しては賛成の意見ばかりがあるのではない．モニタの概念に対しての批判は文献[20]を参照されたい．同期プリミティブはオペレーティングシステムを開発する者にとって基本的な課題である．これらの中で重要な論文を文献[13],[14],[15],[16],[17],[18],[19],[20]に示しておく．

関数型言語.

オペレーティングシステムを構築する試みは関数型言語のコミュニティでも行われている．まず，文献[21],[22],[23]は関数型オペレーティングシステムの提案をしている．実世界で使用できる関数型オペレーティングシステムは未だ実現されておらず，今後に残された問題である．

SML, HASKEL, FX91, and Concurrent Clean.

関数型言語のコミュニティでは並列・並行関数型言語を開発することが活発に行われている．この代表的なものに，New Jersey SML [24], [25], [26], [27], [28], [29], [30], HASKEL[31],[32], FX91[33],[34], Concurrent Clean[35],[36]がある．

並列関数型言語は実際のプログラミングか？

J. H. Morrisは関数型言語は実際のプログラミング(Real Programming)であるということを主張した論文で，一般に関数型言語は「セマンティクスの定義が容易にできるとともに数学的なエレガンスをもっているためにメタプログラマが好んで使う言語である」と考えられていると言及している[39]．しかし，文献[38],[39], [40],[41],[42],[43]は現実の問題を関数型言語で記述している．今後は他の言語との性能比較をすることが必要である．

並列関数型言語で例外処理や割込みはできるか？

関数型言語でユーザレベルからプロセス管理を行うことは可能である．文献[45],[46,[47]は関数型言語が並行処理だけでなく同期操作，非同期操作，例外処理などの機能を継続を使って実現できることを示している．また，文献[48]はSMLでLindaが実現可能なことを示し，並列計算機にも適用可能なことを示している．

関数型言語で並行プログラムを書くと，次に示す利点がある．まず，多相型関数(Polymorphic function)によって，定義された関数をさまざまな用途に使用できること．次に，かならずしも停止するプログラムを用意する必要がないこと．もし，必要がなくなればガーベジコレクションで自動的に回収する．最後に，継続をランタイムスタック無しで実現可能なコンパイラ使用すればより少ないコードのスレッドが実現できる．この理由で，効率のよいプロセス管理を行うことが可能である．もし，他の言語と性能比較できるところまでいけば，J. H. Morrisが主張したかったように，関数型言語が実際のプログラミングになるのもそう遠くはないであろう．

Scheme Community.

同じ関数型言語といってもLispとSchemeは関数型言語のコミュニティとは別のコミュニティで発展を遂げてきた．ActorモデルはLispでモデル化できるはずであるであるという信念を持ったG. L. Steele Jr.が開発したScheme[49]がこれから述べる継続を使った並列・並行処理に重要な役割を果たす．Schemeそれ自身は並行処理を行う機能を持ち合わせていないが，後述する継続を実際の言語にサポートした最初の言語である．SchemeはAlgol 60の流れをくんでいる．その当時，Algol 60を高階プログラミング言語にするという研究がP. J.Landin [50],[51], J. C. Reynolds[52],[53]らにより行われていた．勿論，これらの言語には継続が定義できるようになっている．ただし，これらの言語は実験プログラミング(Experimental Programming)であった．それに反して，Schemeは実際のプログラミングである．一般にLisp/Schemeは遅いという批判があった（現在でもある），そのような誤った批判はGNU LISPを使う人が増えて，Lisp/Schemeは実際のプログラミングであるということが多くの人に理解されて一般的な評価はよくなってきたといえる．

SchemeはMIT, Yale 大学, Indiana大学, ベンダに引き継がれた．Yale大学の Scheme は T Scheme と言われているがCPS(Continuation Passing Style)を使ったコンパイラで驚異的に高速なSchemeである．この手法はNew Jersey SMLのコンパイラにも使用されている．T Schemeの高速コンパイラを作った D. A. Kranz は MIT に移りMul-T という並列Scheme[58]を作った．

さて，Indiana大学のSchemeの研究は注目するに値する．特に，M. Wandは継続を使えば並行処理ができるということを言い出した最初の人である[54]．このシステムはハンドコーディングであるが，オペレーティングシステムの機能であるプロセス管理をSchemeから容易に行うことができるということで，ユーザレベルでプロセス管理を行おうと考えていた多くの研究者を勇気ずけた．

Parallel Lisp and Parallel Scheme Community.

LispとSchemeは実験プログラミングでもメタプログラミングでもない．このため多くの研究機関で比較的早い時期から並列・並行処理が可能なLispとSchemeの開発が行われた．参考のために，著者が知り得た文献を[55]から[64]まで示しておく．

Parallel Processing Community.

並列処理の研究分野ではユーザレベルでプロセス管理を行うという，いわば制御レベルのパラレリズムのみならず，データレベルのパラレリズムをユーザが行うことによって並列性の性能向上を計ることが行われている．このコミュニティは独自のシステムを開発しておりオペレーティングシステムとの関連を気にして

いないのではないかと考えている．

Object-Oriented Languages Community.

オブジェクト指向のコミュニティでも複数のオブジェクトを生成して並行に実行させようとすることを考えている．オブジェクトとプロセスとは独立した概念ではあるが，プロセスはオブジェクトのメソッドを呼び出さなければならないということでプロセス管理に関ってくる．この分野の研究については最近でたサーベイを参照されたい[65]．

マイクロカーネルのオペレーティングシステム．

これまでは，ユーザがプロセス管理をプログラミング言語レベルから行おうとする立場からの研究を紹介した．ユーザのさまざまな要求がオペレーティングシステムに対してされたが，UNIXを含めどのオペレーティングシステムもユーザの厳しい要求には応えなかった．ユーザにとって光明とも見えるオペレーティングシステムが出現したのは'８６年に発表されたMachであった．このMach[66]に刺激されてオペレーティングシステムの開発・研究が盛んに行われるようになった．これらのオペレーティングシステムは一般にマイクロカーネルで構成されていると云われる．このマイクロカーネルの特徴は，従来のモノシリックなオペレーティングシステムとは違い，従来のオペレーティングシステムの機能をいくつかモジュールデ再構成して，それぞれをサーバとして実現するものである．マイクロカーネルをサポートしているMach以外のオペレーティングシステムには[67],[68],[69],[70]がある．

C Threads.

C Threadsを開発したE. C. Cooperも述懐しているようにDEC SRCのTopaz[71]から影響を受けて，C言語からマイクロカーネルとのインタフェースをとるということが考えられた．C Threadsは現在広く使用され，UNIXの標準化の進捗とともに仕様の変更がなされるであろう．C Threadsに関連する論文を[72],[73],[74],[75],[76],[77],[78]に示しておく．

例外処理とその他の処理ならびにスケジューリング．

言語レベルからプロセスの管理をする場合に考えなければならない他の重要な問題である例外処理，プロセス間通信，I/O処理などの問題，ならびにマイクロカーネルによる分散アルゴリズムについては文献[79],[80],[81],[82],[83],[84],[85],[86]，スケジューリングについては文献[87],[88],[89]を参照されたい．

ユーザレベルで行うプロセス管理とカーネルレベルで行うプロセス管理のジレンマ．

ユーザレベルでスレッドを管理する場合にはカーネルの介在は必要ない．その結果，良好な性能が得られる．しかし，もしこれらのスレッドが従来のオペレーティングシステムのプロセスに割り付けられて実行されるとスレッドがCPUを占有する特権は与えられず，その結果，性能はあまり期待できないものになる．マイクロカーネルは複数のスレッドを管理をカーネルレベルで行うことを考えた．しかし，性能はユーザレベルのそれよりは悪くなる．一方，ユーザレベルはカーネルの介在なしで管理を行うのでシステム統合の点では問題がある．このため，最近のマイクロカーネルはユーザレベルのスレッドパッケージにCPUを獲得する情報をupcallで知らせたり，仮想マシンをユーザレベルとカーネルレベルの間に配置し，カーネルもユーザもCPUを直接獲得できるようにするなどの配慮を計っている．この文献は[70],[77],[90],[91],[92]に示しておく．最近の研究は，ユーザレベルとカーネルレベルの協調を重視するようになったのが特徴的である．この動機はユーザレベルへの配慮とリアルタイムへの応用であろう．

継続(Continuations)

継続とは，ある時点以降の計算を表現している関数である．本稿ではすでに数カ所で継続という言葉を使用したが，オペレーティングシステムの言葉で言えばスレッドコンテキストの抽象化であるといってよい．継続に関する文献は[94],[95],[96],[97]に示しておくが，オペレーティングシステムでも継続を使ったものが存在するとともに[93]，プログラミング言語のコミュニティでも継続だけのワークショップを開催するなどの隆盛である[96]．難を言えば，理論的な学会で議論されており，論文を理解するのに苦労するということである．しかし，継続はこれからもいろいろの方向に発展する可能性を秘めており，ユーザレベルでプロセス管理やもっと一般的な制御を考えている研究者は関心を持っていることが望ましいと考えている．

今後のオペレーティングシステムはユーザとどのように関わりあいを持てばよいか．

高機能のプロセッサを搭載したワークステーションでマルチスレッドをサポートしたオペレーティングシステム上で並行プロセスを実際に実行したとき，通常は予想以上の性能の悪さに愕然とするものである．今後，マルチプロッセサワークステーションが普及するであろうが，このような問題はますます起こることであろう．これを解決するためにシステム全体のチュウ

ニング，性能評価やデバッグをオペレーティングシステムが助けることが必要になる[98],[99],[100],[101],[102],[103],[104],[105].

リアルタイム処理のオペレーティングシステムは今後ますます重要になる，特にソフトウェア要求定義におけるリアルタイムの制約定義と分析が実際にオペレーティングシステムでサポートされることが望ましい．

リアルタイム処理はソフトウェア工学の対象となるには問題が山積しており，一番遅れている．しかし，着実に研究は進んでいる．ソフトウェア工学でとりあげられたリアルタイム処理の最近の研究を文献[106]から[108]に示しておく．

最近のユーザレベルとカーネルレベルの協調はこのまま続いてほしい．しかし，ユーザは限りなくコンカレンシイとパラレリズムの夢を追い続けるであろう．しかし，ユーザがすべての計算機資源の情報を知り得ることは不可能である．今後とも，カーネルを通してしか資源の情報を得ることはできない．カーネルはユーザにとって有用であるとともに適切な情報を提供しなければならない．

おわりに

カーネルレベルによるプロセス管理，いわば，オペレーティングシステムを構築する側とユーザレベルでプロセス管理をする，いわば，計算機の性能をフルに発揮せんとする側と関連について私論を展開した．結論としては，現在が一番両者が接近している時代であると考えている．しかしながら，残された問題は多々ある．両者はそれぞれの問題の解決をすることになるであろう．しかし，多くのユーザがマイクロカーネルを使って，その上で新しいコンカレンシイとパラレリズムの研究を展開して欲しい．マイクロカーネルが多くの研究者によって使われることを期待している．

文　献

[1] J. Stoy and C. Strachey, "OS6, An Operating System for a Small Computer.," Oxford Univ. Computing Laboratory, Programming Research Group Tech. Monograph PRG-8, May 1972.
Also published in *Computer. J.*, vol. 15, no. 2 and 3, May and Aug. 1972, pp. 117-124 and pp. 195-203.

[2] C. Strachey and J. Stoy, "The Text of OSPub," Oxford Univ. Computing Laboratory, Programming Research Group Tech. Monograph PRG-9(t), July 1972.
（文献 [1] のソースプログラム BCPL で書かれている）

[3] C. Strachey and J. Stoy, " The Text of OSPub (Commentary)," Oxford Univ. Computing Laboratory, Programming Research Group Tech. Monograph PRG-9(c), July 1972.
（文献 [2] のソースプログラムの注釈）

[4] J. G. Mitchell, W. Maybury, and R. Sweet, "Mesa Language Manual, Version 5.0" Xerox PARC, CSL-79-3, Apr. 1979.

[5] B. W. Lampson and D. D. Redell, "Experience with Processes and Monitors in Mesa," *Comm. ACM*, vol. 23, no. 2, Feb. 1980, pp. 105-117.

[6] B. W. Lampson and R. F. Sproull, "An Open Operating System for a Single-User Machine," in *Proc. Seventh ACM Symp. Oper. Syst. Principles*, Dec. 1979, pp.110-121.

[7] S. Harbison, "Modula-3," *BYTE*, Nov. 1990, pp. 385-388, p. 390, and p. 392.

[8] S. P. Harbison, *Modula-3*. Prentice Hall, 1992.

[9] G. Nelson ed., *Systems Programming with Modula-3*. Prentice Hall, 1991.

[10] L. Cardelli, J. Donahue, L. Glassman, M. Jordan, B. Kalsow, and G. Nelson, "Modula-3 Report (revised)" DEC SRC, Res. Rep. 52, Nov. 1989.

[11] M. Reiser, *The Obelon System, User Guide and Programmer's Manual*. ACM Press, 1991.

[12] M. Reiser and N. Wirth, *Programming in Obelon, Step beyond Pascal and Modula*. ACM Press, 1992.

[13] L. Lamport, "A Simple Approach to Specifying Concurrent Systems," DEC SRC, Res. Rep. 15, Dec. 1986.

[14] L. Lamport, "A Fast Mutual Exclusion Algorithm," *ACM Trans. Comp. Syst.*, vol. 5, no. 1, Feb. 1987, pp. 1-11.

[15] A. D. Birrel, J. V. Guttag, J. J. Horning, R. Levin, "Synchronization Primitives for a Multiprocessor, A Formal Specification," in *Proc. Eleventh ACM Symp. on Oper. Syst. Principles*, Nov. 1987, pp.94-102.

[16] A. D. Birrel, "An Introduction to Programming with Threads," DEC SRC, Res. Rep. 35, Jan. 1989.

[17] B. N. Bershad, "Mutual Exclusion for Uniprocessors," Carnegie Mellon Univ., Tech. Rep. CMU-CS-91-116, Apr. 1991.

[18] B. Lampson, W. Weihl, and E. Brewer, *6.826 Principles of Computer. Systems*. Lecture Notes and Handout, MIT/LCS/RSS 19, July 1992.

[19] B. N. Bershad, D. D. Redell, and J. R. Ellis, "Fast Mutual Exclusion for Uniprocessors," *SIGPLAN Notices*, vol. 27, no. 9, Sep. 1992, pp. 223-233.

[20] A. J. Bernstein and P. M. Lewis, *Concurrency in Programming and Database Systems*. Jones and Bartlett Publishers, 1993 pp. 185-191.

[21] W. Stoye, "Message-Based Functional Operating Systems," *Sci. of Comp. Prog.*, vol. 6, 1986, pp. 291-311.

[22] D. Turner, "Functional Programming and Communicating Processes," Lecture Notes in Computer Science, vol. 259, Springer, pp. 54-74.

[23] D. Turner, "An Approach to Functional Operating Systems," in *Research Topics in Functional Programming*, D. A. Turner ed., Addison-Wesley, 1990

[24] Norman Ramsey, "Concurrent Programming in ML," Princeton Univ., Tech. Rep. CS-TR-262-90, Apr. 1990.

[25] E. C. Copper and J. G. Morriset, "Adding Threads to Standard ML," Carnegie Mellon Univ., Tech. Rep. CMU-CS-90-186, Dec. 1990.

[26] R. Harper, R. Milner, and M. Tofte, "Standard ML," Univ. of Edinburgh LFCS Rep. Ser. ECS-LFCS-86-2, Mar. 1986.

[27] R. Harper, "Introduction to Standard ML," Univ. of Edinburgh LFCS Rep. Ser. ECS-LFCS-86-14, Revised ed., Jan. 1989.

[28] M. Tofte, "Four Lectures on Standard ML," Univ. of Edinburgh LFCS Rep. Ser. ECS-LFCS-89-73, Mar. 1989.

[29] R. Milner, M. Tofte, and R. Harper, *The Definition of Standard ML*. The MIT Press, 1990.
Also published as R. Harper, R. Milner, M. Tofte, "The Definition of Standard ML, Version 3," Univ. of Edinburgh LFCS Rep. Ser. ECS-LFCS-89-81, May 1989.

[30] R. Milner and M. Tofte, *Commentary on Standard ML*. The MIT Press, 1991.

[31] J. H. Fasel, P. Hudak, S. Peytone-Jones, and P. Wadler, *SIGPLAN Notices Special Issue on the Functional Programming Language Haskell. ACM SIGPLAN NOTICES*, vol. 27, no. 5, May 1992.

[32] A. J. T. Davie, *An Introduction to Functional Programming Systems Using Haskell*. Cambridge Univ. Press, 1992.

[33] D. K. Gifford, P. Jouveloy, M. A. Sheldon, and J. W. O'Toole, "Report on the FX-91 Programming Language," MIT/LCS/TR-531, Feb. 1992.

[34] D. Grundman, R. Stata, and J. O'Toole, "μ FX/DLX A Pedagogic Compiler," MIT/LCS/TR-538, Mar. 1992.

[35] M. van Eekelen, H. Huitema, E. Nocker, S. Smetsers, and R. Plasmeijer, "Concurrent Clean. Language Manual-Version 0.8," Univ. of Nijmegen Tech. Rep. no. 92-18, Aug. 1992.

[36] H. Huitema and Rinus Plasmeijer, "The Concurrent Clean System User's Manual & PABCstat User's Manual," Univ. of Nijmegen Tech. Rep. no. 92-19, Aug. 1992.

[37] J. H. Morris, "Real Programming in Functional Languages," in *Functional Programming and its Applications*, edited by J. Darlington, P. Henderson, and D. A. Turner, Cambridge Univ. Press, 1982.

[38] P. Hudak and S. Anderson, *Haskell Solutions to the Language Session Problems at the 1988 Saliahan High-Speed Computing Conference*. Yale Univ., Res. Rep. YALEU/DCS/RR-627, May 1988.

[39] Y. Kashiwagi and D. S. Wise, "Graph Algorithms in a Lazy Functional Programming Language," Indiana Univ. Tech. Rep. no. 330, Apr. 1991.
（この論文は通常の関数型言語で実現しているが並列関数型言語で実現できる可能性を持っている．）

[40] P.W.M. Koopman, M.C.J.D. van Eekelen, and M.J. Plasmeijer, "Functional Description of Neural Networks," Proc. Int. Neural Network Conf., 1990, pp. 701-704.

[41] Pieter Koopman, *Functional Programs as Executable Specifications*. Ph. D. Thesis, Univ. of Nijmegen, 1990.

[42] L.M.W.J. Rutten, P.W.M. Koopman, M.C.J.D. vanEekelen, and M.J. Plasmeijer, "Inventory and Functional Prototypes of Neural Networks," Univ. of Nijmegen, Tech. Rep. no. 90-15, Oct. 1990.

[43] L.M.W.J. Rutten, P.W.M. Koopman, M.C.J.D. van Eekelen, and M.J. Plasmeijer, "Functional Specification of a Neural Network," in Proc. ICANN-91, June 1991.

[44] J. H. Reppy, "First-class synchronous operations in Standard ML," Cornell Univ., Tech. Rep. TR 89-1068, Dec. 1989.

[45] J. H. Reppy, "Asynchronous Signals in Standard ML," Cornell Univ., Tech. Rep. TR 90-1144, Aug. 1990.

[46] J. H. Reppy, "Synchronous operations as first-class values," in *Proc. SIGPLAN '88 Conf.on Progr. Lang.Design and Implementation*, June 1988, pp. 250-259.

[47] R. Govindarajan, "Object Shielding: A Fresh Look at Exception Handlers in Parallel Functional Languages," in *Proc. of 1992 IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems*, July 1992, pp. 158-165.

[48] E. H. Siegel and E. C. Cooper, "Implementing Distributed Linda in Standard ML," Carnegie Mellon Univ., Tech. Rep. CMU-CS-91-151, Oct. 1991.

[49] G. L. Steele, Jr. and G. J. Sussman, "The Revised Report on SCHEME," MIT Art. Intell.Memo no. 379, Jan. 1978.

[50] P. J. Landin, "A Correspondence between ALGOL 60 and Church's Lambda-Notation: Part I and Part II," *Comm. ACM*, vol. 3, no. 2 and 3, Feb. and Mar. 1965, pp. 89-101 and 158-165.

[51] P. J. Landin, "The Next 700 Programming Languages"*Comm. ACM*, vol. 9, no. 3, Mar. 1966, pp. 157-166.

[52] J. C. Reynolds, "GEDANKEN-A Simple Typeless Language Based on the Principle of Completeness and the Reference Concept," *Comm. ACM*, vol. 13, no. 5, May. 1970, pp. 308-319.

[53] J. C. Reynolds, "Definitional Interpreters for Higher-Order Programming Languages," in Proc. ACM National Conf., 1972, pp. 717-740.

[54] M. Wand, "Continuation-Based Multiprocessing," in Proc. 1980 LISP Conf., 1980, pp. 19-28.

[55] R. H. Halstead, "Multilisp, A Language for Concurrent Symbolic Computation," *ACM Trans. Prog. Lang. and Syst.*, vol. 7, no. 4, Oct. 1985, pp. 501-538.

[56] S. Miller, "MultiScheme, A Parallel Processing System Based on MIT Scheme," MIT, Tech. Rep., MIT/LCS/TR-402, Sep. 1987.

[57] A. Rabinov, I. Rivin, "Programming in Qlisp-A Case Study," Stanford Univ. CS Rep. STAN-CS-89-1247, Feb. 1989.

[58] D. A. Kranz, R. Halstead, E. Mohr, "Mul-T, A High-Performance Parallel Lisp," in *ACM SIGPLAN '89 Symp. on Prog. Lang. Design and Implementation*, June 1989, pp. 81-90

[59] J. S. Weening, "Parallel Execution of Lisp Programs," Stanford Univ. CS Rep. STAN-CS-89-1265, June 1989.

[60] T. Ito and R. H. Halstead, Jr. eds., *Parallel Lisp, Languages and Systems.* Lecture Notes in Computer Science, vol. 441, Springer, 1990.

[61] T. Ito and M. Masui, "A Parallel Language PaiLisp and its Kernel Specification," in [60] pp. 58-100.

[62] C. L. Wang, "A Continuation-Based Language Embedded In Scheme," *Comput. Lang.* vol. 17, no. 1, pp. 19-37, 1992.

[63] T. Ito and T. Seino, "On PaiLisp Continuation and its Implementation," in [96], pp. 73-90.

[64] S. Jagannathan and J. Philbin, "A Foundation for an Efficient Multi-Threaded Scheme Systems," in *Proc. ACM Conference on Lisp and Functional Programming*, June 1992, pp. 345-357.

[65] B. B. Wyatt, K. Kavi, and S. Hufnagel, "Parallelism in Object-Oriented Languages: A Survey," *IEEE Software*, vol. 9, no. 6, Nov. 1992, pp. 56-66.

[66] M. J. Accetta, R. V. Baron, W. Bolosky, D. B. Golub, R. F. Rashid, A. Tevanian, Jr., M. W. Young, "Mach, A New Kernel Foundation for UNIX Development," in *Proc. of Summer USENIX Conf.*, Jun. 1986, pp. 93-113.

[67] D. R. Cheriton, G. R. Whitehead, and E. W. Sznyter, "Binary Emulation of UNIX using the V Kernel," in *Proc. of Summer USENIX Conf.*, June 1990, pp. 87-96.

[68] S. J. Mullender, G. van Rossum, A. S. Tanenbaum, R.van Renesse, and H. van Staveren, "Amoeba, A Distributed Operating System for the 1990s," *IEEE Comp. Mag.*, vol. 23, no. 5, May 1990, pp. 44-54.

[69] M. Guillemont, J. Lipkis, D. Orr, and M. Rozier,"Chorus systems, A Second-Generation Micro-Kernel Based UNIX; Lessons in Performance and Compatibility," in *Proc. of Winter USENIX Conf.*, Jan. 1991, pp. 13-22.

[70] M. Gien, "From Operating Systems to Cooperative Operating Environments," in *Proc. 10th Anniversary Int. UNIX Symp.*, July 1992, pp. 1-10.

[71] P. R. McJones and G. F. Swart, "Evolving the UNIX System Interface to Support Multithreaded Programs," Res. Rep. 21, DEC SRC, Sep. 1987.

[72] E. C. Cooper, Richard P. Draves, "C Threads," Carnegie Mellon Univ., Tech. Rep. CMU-CS-88-154, June 1988.

[73] UNIX International, *Multiprocessing Work Group, Final Report.* UNIX International, Inc., 1989.

[74] Open Software Foundation, *Application Environment Specification (AES), Operating System Programming Interface Volume.* Prentice Hall, 1990.

[75] IEEE Comput. Society, *Treads Extension for Portable Operating Systems.* Technical Committee on Operating Systems, Draft P1003.4a/D5, Dec. 1990.

[76] M. B. Jones, "Bringing the C Libraries with us into a Multi-Threaded Future," in *Proc. of Winter USENIX Conf.*, Jan. 1991, pp. 81-91.

[77] M. L. Powell, S.R. Kleiman, S. Barton, D. Shah, D. Stein, M. Weeks, "SunOS Multi-thread Architecture," in *Proc. of Winter USENIX Conf.*, Jan. 1991, pp. 65-79.

[78] G. Cattaneo, G. Di Giore, M. Ruotolo, "Another C Threads Library," *SIGPLAN*, vol. 27, no. 12, Dec. 1992, pp. 81-90.

[79] D. D. Clark, "The Structuring of Systems Using Upcalls," in *Proc. Tenth ACM Symp. on Oper. Syst. Principles*, Dec. 1985, pp.171-180.

[80] E. S. Roberts, "Implementing Exceptions in C" Res. Rep. 40, DEC SRC, Mar. 1989.

[81] D. L. Black, D. B. Golub, K. Hauth, A. Tevanian, and R. D. Sanzi, "The Mach Exception Handling Facility," Carnegie Mellon Univ., Tech. Rep. CMU-CS-88-129, Apr. 1988.

[82] M. D. Schroeder, M. Burrows, "Performance of FireflyRPC," in *Proc. Twelfth ACM Symposium on Operating Systems Principles*, Dec. 1989, pp. 83-90.

[83] B. N. Bershad, T. E. Anderson, E. D. Lazowska, and H. M. Levy, "Lightweight Remote Procedure Call," in *Proc. Twelfth ACM Symposium on Operating Systems Principles*, Dec. 1989, pp. 102-113.

[84] A. Forin, D. Golub, and B. N. Bershad, "An I/O System for Mach 3.0," Carnegie Mellon Univ., Tech. Rep. CMU-CS-91-191, Oct. 1991.

[85] M. Vandevoorde and E. S. Roberts, "WorkCrews, An Abstraction for Controlling Parallelism," *Int. J. of Parallel Programming*, vol. 17, no. 14, Aug. 1988, pp. 347-366.

[86] H. Bal, *Programming Distributed Systems.* Prentice Hall, 1990.

[87] D. L. Black, "Scheduling Support for Concurrency and Parallelism in the Mach Operating System," *IEEE Comput. Mag.*, vol. 23, no.5, pp. 35-43, May 1990.

[88] D. L. Black, "Scheduling and Resource Management Techniques for Multiprocessors," Carnegie Mellon Univ., Tech. Rep. CMU-CS-90-152, July 1990.

[89] D. L. Black, "Processors, Priority, and Policy, Mach Scheduling for New Environments," in *Proc. of Winter USENIX Conf.*, Jan. 1991, pp. 1-12.

[90] K. Loepere, *Mach 3 Kernel Principles, MK67.* OSF and CMU, Jan. 1992.

[91] T. E. Anderson, B. N. Bershad, E. D. Lazowska, H. M. Levy, "Scheduler Activations, Effective Kernel Support for the User-Level Management of Parallelism," in *Proc. Eleventh ACM Symp. Oper. Syst. Principles*, October 1991, pp. 95-109.

[92] B. D. Marsh, M. L. Scott, T. J. Leblanc, E. P. Markatos, "First-Class User-Level Threads," in *Proc. Eleventh ACM Symp. Oper. Syst. Principles*, Oct. 1991, pp. 110-121.

[93] R. P. Draves, B. N. Bershad, R. F. Rashid, R. W. Dean, "Using Continuations to Implement Thread Management and Communication in Operating Systems," in *Proc. Eleventh ACM Symp. Oper. Syst. Principles*, October 1991, pp. 122-136.

[94] G. F. Johnson and D. Duggan, "Stores and Partial Continuations as First-Class Objects in a Languages and its Environment," in *Proc. Fifteenth Annual ACM Symp. on Principles of Prog. Lang.*, Jan. 1988, pp. 158-168.

[95] C. Strachey and C. P. Wadsworth, "Continuations - A mathematical Semantics for Handling Full Jumps," Oxford Univ., Programming Research Group Technical Monograph PRG-11, 1974.

[96] O. Danvy, C. Talcott eds, *Proc. of the ACM SINGPLAN Workshop on Continuations CW92.* Published as Stanford Univ. Tech. Rep. STAN-CS-92-1426, June 1992.

[97] W. D. Clinger, A. H. Hartheimer, E. M. Ost, "Implementation Strategies for Continuations," in *Proc. 1988 ACM Conference on Lisp and Functional Programming*, July 1988, pp. 124-131.

[98] T. Lehr, D. Black, Z. Segall, D. Brsalovic, "MKM, Mach Kernel Monitor Description, Examples and Measurements," Carnegie Mellon Univ., Tech. Rep. CMU-CS-89-131, Mar. 1989.

[99] Frank Lange, Reinhold Kroeger, Martin Gergeleit, "JEWEL, Design and Implementation of a Distributed Measurement System," *IEEE Trans. Parallel and Distributed Syst.*, vol. 3, no. 6, Nov., 1992. pp.657-671.

[100] H. Tokuda and M. Kotera, "A Real-Time Toolset for the ARTS Kernel," Carnegie Mellon Univ., Tech. Rep. CMU-CS-88-180, 1990.

[101] H. Tokuda, M. Kotera, Clifford W. Mercer, "A Real-Time Monitor for a Distributed Real-Time Operating System," Carnegie Mellon Univ., Tech. Rep. CMU-CS-88-179, 1988.

[102] C. W. Mercer, Y. Ishikawa, and H. Tokuda, "Distributed Hartstone Real-Time Benchmark Suite," Carnegie Mellon Univ., Tech. Rep. CMU-CS90-110, Mar. 1990.

[103] N. Islam and R. H. Campbell, "Design Considerations for Shared Memory Multiprocessor Message Systems," *IEEE Trans. Parallel and Distributed Syst.*, vol. 3, no. 6, Nov., 1992. pp. 702-711.

[104] T. E. Anderson and E. D. Lazowska, "Quartz, A Tool for Tuning Parallel Program Performance," in *Proc. 1990 ACM SIGMETRICS Conf. Measurement and Modeling of Comput. Syst.*, May 1990, pp. 115-125.

[105] D. Caswell, D. Black, "Implementing a Mach Debugger for Multithreaded Applications," in *Proc. of Winter USENIX Conf.*, Jan. 1990, pp. 25-39.

[106] A. Shaw and K. Jeffay, "Software Engineering of Real-Time Operating Systems," Univ. of Washington, Tech. Rep. 88-0101, Jan. 1988.

[107] Luqi, "Real-Time Constraints in a Rapid Prototyping Language," *Comput. Lang.* vol. 18, no. 2, 1993, pp. 77-103.

[108] J. P. Calvez, *Embedded Real-Time Systems: A Specification and Design Methodology.* John Wiley & Sons, 1993