III　操作システムまわりの話題から†

和　田　英　一††

1. はじめに

これは情報処理学会 20 周年記念特集号に掲載されるのだそうである．この春駒場から本郷に進学してくる学生諸君の生まれた頃に学会ができたといえば，ずい分昔のことのようにも思うが，その頃卒業していたものからみればついこのあいだのような気もしないでもない．20年というからには，10年前に10周年になったはずだが，その時のことを憶えている方はどのくらいあろうか．あの時学会は 10 周年記念論文を募集した．今回も 20 周年記念論文を公募したが今回は別に課題は与えられなかったようである．ところが前回のは与えられた課題がふたつあって，Aは「ソフトウェア危機の克服」，B は「ヒューリスティックプログラム」であった．ついでながら賞金は課題ごとに1編選んで，選ばれれば5万円ということになっていた．10年たった今回はオーストラリアまでの旅費と IFIP 80 の滞在費がいただけるのだから，ずい分豪勢になったものだと思う．

さて問題は 10 周年記念論文はどうなったかということなのだが，結果が情報処理に発表になったという記憶はない．ただ，情報処理1971年6月号384ページをみると創立 10 周年を記念してもうけられた論文賞規定に基づき第1回の表彰論文の選定を行った．(論文賞選定委員長大泉副会長，他委員20名，第9回通常総会で表彰の予定.)とあり，同382ページの第9回通常総会の記事には，昭和45年度論文賞が坂井利之君（京大）他4名，2件に授与されたが，……とあるだけである．誤解を招くといけないので断っておくが，ここで当時の学会が論文の発表をうやむやにしたと騒ぎたてようとしているのでは決してない．むしろここでいいたいのは，当時としてはやはりこのふたつの課題は荷が重かったのではないか，そしてこのテーマに一生懸命とりくむには当時としては5万円では魅力が少なかったのではないかということである．

ヒューリスティックプログラムの方は別として，ソフトウェア危機は，当時は，いわれはじめて程ないころで，特に操作システムのもたもたが，その危機感を一層に煽ったものと思われる．情報処理 1969 年 11 月号には当時の会長の高橋秀俊先生がソフトウェア危機を警告されている[1]．高橋先生はその中で，「最も注目を浴びている MIT の Multics がまだ本格的にはたらいていないということである．このようなソフトウェアのつまづきは，今後ますます大型化して行こうとする，電子計算機システム自体の将来に，暗いかげを投げかけるものである」といわれた．そして「このような状況は，……，ソフトウェアのもつ本質的な問題につながるように思えるのである」とつづけられた．

しかし 20 周年の今日，事情がずい分ちがってきていると感じるのは筆者だけではないと思う．もちろん操作システムはまだ低い雲におおわれているけれど，ずい分うす日もさしはじめたという感じがする．となり村のプログラム言語の処理系の方は，しばらく前に雲が切れて，かなり景色がはっきりしてきている．これはやはり Algol 60 からはじまったプログラム言語の記述の努力と，それに基づいた処理系の試作が，プログラム言語とその処理系の原理とでもいうべきものを明確にしてきたからに違いない．その結果，処理系の移植などもかなり当り前のことになって，処理系まわりの話題は今やソフトウェア工学のそれになったといってよい．それにくらべれば，操作システムの状況は，たとえば OS 6 の論文[2),3)]のはじめの方に書いてあったことだが，多くのシステムはアドホックに設計されており，それに対して原理を見い出すためには，見通しのよい操作システムを設計し，吟味してみなくてはならなかったし，見通しのよい操作システムを設計するためには，操作システムの記述できるプログラム言語がなくてはならなかった．そういう次第で原理への道は多少遠かったけれども，このごろようやく目

† Topics on Operating Systems by Eiichi WADA (Department of Mathematical Engineering, University of Tokyo).
†† 東京大学工学部計数工学科

的地が視界に入ってきたような気がするのである．今回はそういうわけで，この道程をふりかえってみることにしたい．

2. Multics の終焉

10年前にまだ本格的に働いていなかった MIT の Multics[4),5)] は，この10年間に MIT としては研究のピリオドをうってしまった．その最終報告は MIT の LACS（昔のプロジェクト MAC のこと，Laboratory for Computer Sciences の略）から出版されて[6),7)]，Final Repart of the Multics Kernel Design Project という題である．この題名からみると，核部分の最終報告のようにも見えるが，これで Multics 全体の研究も終ったことになっている．

さてこの白鳥の歌によれば，Multics 最後の研究は，次のようなものであった．Multics はもともと情報保護体制の確実なシステムということで開発がすすみ，よく知られているようにいろいろと斬新な方式を考案してはとり入れて作られてきた．だから他のシステムにくらべれば，断然強固 (secure) なはずであったが，それでもいろいろ問題はあった．一番重要なスーパーバイザは，開発期間を通じて100人以上のプログラマで作られ 54,000 行のソースコードからできているので，誰にもよくはわからず，とても検査などできたものではないしろものになっていた．またこれまで提案された保護機構も，多少アドホックであって，単一の基本モデルになりえず，たとえ誰かが 54,000 行に目を通したところで，何に頼って判断を下してよいかがわからないということであった．そこでMulticsのグループは最後の力を振りしぼって，スーパーバイザのうち，特に核といわれる部分の作り直しにとりかかった．1974年のことである．いいわすれたが，もち論このとき Multics は本格的に働いていた．もっとも学生が絶えずシステムをいじるものだから，よくダウンもしたが，そういうダウンだから割合い短時間で復旧していたように記憶している．

この作り直しは，アセンブラで書いてある部分をできるだけ PL/I で書くようにしたこと，必ずしも核内になくてもよい部分を，核外に出すように努めたこと，抽象的データ型のようにデータの処理を専用の手続きにまかせ，階層状に構成しようとしたことなどに目標がおかれていた．階層状の構成を考えた場合，ふたつの機構がお互いに相手の機構に依存していると困難が生じる．事実 Multics では，最近の操作システムに必ず登場するふたつの基本的機構，つまりプロセスの多重化と仮想記憶方式（プロセス制御とページ制御）が相互依存していて，まず問題となった．もう少し具体的にいうと，ページフォールトがおきると別のプロセスに移ってページを読み込む．一方プロセスの管理表が大きくなってスワップアウトされていたりするとページフォールトがおきるという具合いである．これは丁度，いずれも外からよばれるふたつの手続きpとqが，それぞれ相手を下請けにするようなもので，Pascal[8)] では一方に forward 宣言をつけて解決するが，もし forward 宣言がなければ（相互依存できなければ），どちらかを二重にして対処しなければならないという事情に似ている．結局 Multics ではプロセス制御を二重にした．つまりページ制御を必要としない簡単なプロセス制御を作って，それを一番下請けにし，それをページ制御が使い，それを汎用のプロセス制御が使うという形になった．これでページ制御とプロセス制御にまつわる原理の一部が明らかになったとみることができる．

このような新しい考察に基づいた構成，高水準プログラム言語による記述，核に収めなければならないものの選定とその理由づけを行うことによって，見通しがよくなり，核の中のプログラムの行数も 54 K から 28 K へ減少した．この改良は別に実行効率の向上をめざしたものではなかったが，これによって効率がおちるようでは問題である．結果的には速くなったモジュールあり，反対に遅くなったのもありで，あまり大勢に影響なしということであった．

3. OS 6 と Solo

Multics の最後のプロジェクトより前になるけれども，オックスフォード大学で Strachey と Stoy が開発した OS 6[2),3)] は全体がプログラム言語 BCPL[9)] で記述してあるというので画期的であった．これより前，小さい操作システムを小さいグループで作ってみたという話は，たとえば Dijkstra の THE[10)] や，またたとえば Liskov の Venus[11)] など，報告はあったものの，これらは主要な設計者はひとりであっても，プログラム自体はアセンブラで書かれていたようである．いつだったか Dijkstra が来日した際に，THE を読んでみたいけれど，リストはみせて貰えないかときいてみたら，あれはアセンブラでごたごた書いてあって，とても素人に読めるようなものではないという返事であった．

これに対して OS 6 はプログラムテキストをみんなに読んで貰い，本当にできるということを信用して貰いたいということで，OS 6 のテキストと注釈を発行した[12),13)]．このとき OS 6 とは多少ちがうところがでてきて，OSPub という名前になったけれども，BCPL のことを知っていると，OS 6 がどうなっていたか，解読できるのである．筆者は先年，大学院の諸君とこれを全部読んでみて，やはり実に面白かったことを覚えている．BCPL で書いたといっても，記憶装置の管理はすべて OS 6 が自分で行っている．Stoy らにいわせると，操作システムのためのプログラム言語は，制御構造をサポートすることが特に大切で，プログラム言語が記憶場所のわりつけまでするのはやりすぎのきらいがあるとのことであった．OS 6 の論文を読むと，入出力の取り扱い，特にストリーム関数（論理入出力装置を入力パラメタ，結果型とする関数）の考え方があまりわかり易くは書いてないが，テキストを読むとそれがとてもよく理解でき，大変面白いものであることがわかった．面白い考えであることは確かなのだが，大変遅いこともあって，その後，このようなユニークな考え方のものに出会わないのが残念である．

OS 6 はファイルシステムまで持つシングルユーザ用シングルプロセスの操作システムである．BCPL で処理した再配置可能なオブジェクトプログラムのローダやファイルのカタログ探索の機能などをもっている．操作システムといえば，前章にのべたような多重プロセスやページング機構がついているのが望ましいが，OS 6 を読むと，そのような拡張はさほど困難ではないのではないかと思われる．いずれにしても，操作システムのプログラムも全部読めるものだという認識を持ち得たのは大きな成果であった．

ページングはやらないけれど，多重プロセスを実現し，しかも高水準のプログラム言語で書いたシングルユーザ用操作システムとして，Brinch Hansen の Solo が登場したのは 1976 年であった[14)~17)]．これは多重プロセスのプログラムを書くために Pascal に変更を加えて作ったプログラム言語 Concurrent Pascal で記述されている．Solo の論文には Solo のうち Concurrent Pascal で書かれた部分がほとんど掲載されているので，Solo の中核部分はやはり全部読むことができた．ただこれはプログラムのスタイルの違いなのだが，Solo の方はどのブロックにも似たような名前が使われていたりするので，OS 6 にくらべて，言語のレベルは高い筈なのに読みにくかったという印象であった．一方の OS 6 のテキストは，プログラムのスタイルにも是非注目してほしいといっているだけあって，大変読み易かった．しかしこのふたつをくらべて，どちらかを移植しようとするならば，それは Solo の方であった[18)]．Solo のシステムの一部は Sequential Pascal で書かれているが，我々はすでに Sequential Pascal は移植して持っていたし，BCPL の方は，第一段の移植（中間コードを解釈実行する方式の移植）はすんで，試用はできるようになっていた[19)]が，第二段の移植（計算機の命令を直接実行する方式の移植）は実施していなかったので，実用にはならない状態であった．また，もちろん Solo の方が後発なので，移植のしやすさをはじめいろいろ面白い考えがとり入れてあり，移すのなら Solo だということになった．これもいいわすれたけれど，Solo は Concurrent Pascal マシンというインタプリタの上で走る比較的めずらしい操作システムであったが，それは移植を目的のひとつにしたからであった．そしてこのインタプリタの PDP-11 用のリストも入手することができたので，移植はその気になればすぐにできたのである．インタプリタ方式といえば，OS 6 もそういう操作システムであったから，OS 6 と Solo とだけ問題にしているこの章でいえば，めずらしいどころか両方ともインタプリタなのである．OS 6 はそれを開発した Modular I 計算機のアーキテクチャがあまりにも BCPL のモデルと相容れないので，一皮かぶせたということになっている．そういう操作システムなら，第一段の移植の BCPL があれば間に合うではないかともいわれそうだが，OS 6 はインタプリタに対してもいくつかの要求をもっていたので，ケンブリッジの BCPL とオックスフォードの OS 6 はすぐには結合できないであろうと思われる．

4. Unix

ベル研究所は，Multics のプロジェクトがはじまった頃は，MIT や GE と協力態勢にあったように聞いているが，比較的早くに Multics から手をひき，マルチ（多）をユニ（単）にとりかえて Unix のプロジェクトを推進した．今では Unix は PDP-11 系のための操作システムのように考えられているが，1969 年に開発がはじめられた時は PDP-7 が使われていた．1973 年の SOSP で発表され，1974 年 7 月の Comm. ACM にのった論文[20)]が Unix に関するほとんど最初

のものであった．その後，数式を写植機で構成する話[21),22)]や，プログラム書法[23)]やソフトウェアツール[24)]の本，プログラマワークベンチの話題[25),26)]や，使ってきた人の経験談[27),28)]で Unix の評判はわが国でもだんだん高まり，今では国内でもミニ Unix など，数カ所で動いているようである．Unix に関する総合的な報告は，2年ほど前の BSTJ にある[29)]し，さほど一般的な文献ではないが IFIP WG 2.7 の機関誌にも掲載された[30),31)]ことがある．

Unix の特徴は何かといえば，ひとによって列挙するものが異なるかもしれないが，ひとつにはコマンドが機能の割に単純だということがあるであろう．大型の操作システムのコマンドは常に荘重であるが，Unix のはこれでよいのだろうかと思うほど簡単である．しかもパイプラインといって，コマンド間のファイル名を省略して書くものだから，ますます短いうえに強力なことができるという印象をうける．こういうことができるのは，利用者がベル研内のレベルの高いクラスに限られているからと思われる．

ふたつ目には，階層構造のファイルシステムをもっていて，保護機構をかけていながら，それが結構簡単な方式で実現されているということである．Multics の保護機構を昔々の大将格の鎧とすれば，Unix のは足軽の具足程度かもしれないが，本質的なところは押えているから，例えばこんな程度でも Moo のゲームの得点表の管理が完全にできると最初の論文に述べられていたのは印象的であった．

システムはC言語[32)]で書いてあるので（すべてではないが）Cの処理系のあるところへは移植できるようになったというのも特徴のうちかもしれないが，これはむしろ他の言語の後塵を拝しているというべきか．むしろ Unix, Miniunix, Lsx など，ホストの機械の大きさに応じていろいろのレベルのシステムが用意されている点は，これまで述べたどの操作システムにもなかったことである．Lsx[33)] は LSI-11 用で，ずい分小さくなったと思ったのも束の間，マイコン用に設計された Chaos[34)] という自称 Unix 属のものまで現れるに至った．Chaos は Basic で書いてあるそうである．ついでだが Chaos で唯一面白いのは，これは数人の利用者で CPU を時分割するのだが，その際，主記憶の切換えと保護を IC メモリのチップセレクトでハード的に実施することで，これを読んだときはさすがマイコン時代と感じたことであった．

Comm. ACM に最初の Unix がのってから6年たったけれども，最近でも Unix 関係の論文は時折見受けられる．そのひとつはオーストラリアのニューサウスウェルズ大学で Unix を使った経験談[35)]であり，もうひとつは Unix の検証に関するもの[36)]である．それらにここでたち入る必要はないであろう．

このように話題の豊富な Unix の功績はなんであろうか．それは操作システムもこんなに使い易く，透明に構成することができるという事実を世に示したことである．

Unix の話題から去る前に筆者の Unix に対する感想をちょっとだけ述べさせて貰いたい．Unix がある意味で，あるいはかなりの意味でよくできている操作システム（プラスプログラムの集積）であることは重重みとめるし，あんなシステムを作ってみたいとも思うけれど，一方その人間とのインタフェースにはなんとなく好きになれない点もどこかにあるような気もする．そう目くじらをたてるほどのことではないかもしれないけれども，ひとつにはその特徴でもある ls というようなコマンドの形があまりにも短いこと．もしかしたら，最短符号化をやっているのではないかと疑いたくなる程で，所外にも売りだすシステムの設計としてはそこまでしなくてもよかったのではないかという気もする．この傾向はBとかCとかいうぶっきらぼうなプログラム言語名にも反映されている．もうひとつには，かの有名なパイプラインのコマンドだが．pr<f1>f2 のように独立したコマンドでは入力出力パラメタとも右へ置くものが，……|sort|pr のように一旦パイプライン中のコマンドとして使われると途端に入力がコマンド名の左にあるように振舞うことである．そのくらいなら独立して使われるときも入力を左に書けばよさそうに思うのだが，この辺は水かけ論になりそうだから先へ進もう．

5. Thoth, Tripos, Muss

ここに並べた不思議な名前は，どれも操作システムである．Thoth[37)] はカナダのウォータールー大学で，Tripos[38)] はイギリスのケンブリッジ大学で，そして Muss[39)] はイギリスのマンチェスター大学で開発された．並べたからには共通点がある筈で，

i）どれも名前の意味がさっぱり分らない．もっとも Thoth については論文の最後に解説がついていた．それによるとエジプト神話にでてくる文字，学問の神で，エジプト人は死ぬと Thoth がその心臓の重量から有罪か無罪かをきめるとあるので閻魔ともいうべき

操作システムである．Tripos の終りの方は portable operating system の意か，そして Muss のMはマンチェスターであろう．それにしても名前がむずかしくなり，またその意味もあまり詮索されなくなってきたものである．

ii）どれも昨年，雑誌に発表された．Muss などはずい分前からやっていたようであるが，筆者はたまたま参考文献に引用したものでほんの数カ月の幅の中で次々と知った．この原稿のはじめの方に書いたように，操作システムにもそろそろ転換期がきたと筆者に思わせたのは，実はこのみっつのシステムのきびすを接しての発表であった．

iii）どれもシステムの移植ということに重点をおいている．これも操作システムの「雪解け」を示唆していないだろうか．

iv）どれもプロセス間の同期にメッセージの受け渡しを採用している．

ことにまず気がつく．もち論あとのふたつが本質的である．話をその点に限れば Solo だって仲間みたいなものだともいえるが Solo は本稿ではすでに退場しているのでここでは遠慮願う．

移植に関していえば，移植するかしないかの選択になるが，今後，操作システムをソフトウェア工学の対象とするためには移植を考えているというのがデフォールトな答になるであろう．ただし Thoth が Thoth ドメインという表現で，移植も無制限にはできないと断っているが，今後はそのようなドメインを何らかの形で付記することになろう．つまりどのドメインなら移植の難易度どの程度というように．

単一プロセスの操作システムというのも今後は一人前とは見られなくなろうから同期もどうとるか，やはりどのシステムでも態度の決定を迫られる．ちなみにもうひとつの方式は共有変数と p,v 操作などを使うものだが，その方が歴史的には早くから論じられていたのに現時点ではメッセージ方式が流行のようである．Muss の論文にも書いてあったことだが，ふたつのプロセスが別の CPU にのっていればたしかにメッセージ方式の方が考え易いかとも思う．しかし，Brinch Hansen の「オペレーティングシステムの原理[40]」を読んだときの，メッセージ方式では先方のプロセスが死んでいた場合の配慮がなんともわずらわしく思われたことを想起してしまう．これも趣味の問題といえばそれまでだが，この両方式の比較はかつてケンブリッジ大学の Needham らによってなされた[41]ことを書き加えておく．多少突飛な考えかもしれないが，Lisp の shallow binding（変数の値を変数アトムの値セルにおく方式）と deep binding（変数の値を環境を構成する a リストにおく方式）を連続移行させることに成功した Baker のような工夫[42]ができればそれは面白いことになるとも思う．

共通点はこのくらいにして，それぞれのシステムの目的には異なる部分もある．Muss では自分たちのまわりの操作システムを全部これでおきかえてしまいたいと考えているようである．したがって Muss の移植と同時に Muss がサポートする言語の処理系をすべて移植するということをやっているみたいである．Thoth はどちらかといえば，実時間の応用プログラムをサポートするのに重点をおいているので，コマンド言語に関する記述は論文にはない．システム自体は BCPL 系の言語で書かれているよし．前二者にくらべると Tripos は個人用計算機の操作システムを念頭において出発した．システムの記述言語は BCPL（のコンカレント版？）である．Muss の記述言語はさだかでない．

6. システム記述言語

現実がどうなのかはわからないが筆者の視野に映る範囲では，操作システムは殆ど高水準言語で書かれるようになった．このはしりはやはり PL/I あるいは EPL で書いた Multics かとも思うが，MIT の Corbató が，当時の事情を説明したもの[43]の最後の方で，もしもう一度 Multics を作ることになったら，時間がたっぷりあれば記述用の言語の設計からはじめたい．だが反対に時間があまりなければ，多分 Fortran でいくだろうと述べているのは面白い．PL/I のポインタ変数が諸悪の根源だという理由と思う．その MIT の Multics のもとへ Richards がきて BCPL を作りケンブリッジ大学へもって帰るわけだが MIT もケンブリッジにあるのだから，BCPL はずうっとケンブリッジで育った言語といってよい．次に PL 360 や Bliss が登場する順かと思うが文献ごと省略する．

PL/I 失格後，型有り言語でシステム記述言語のような顔もさせられたのは Pascal で，目下検討されている国際規格案にも「今やシステム記述言語で……」という語句があるが，これは大問題．筆者は先頃の木下の意見[44]に全面的に賛成である．たしかにひとつには Pascal の処理系が Pascal で書いてあるということもあり，またページ制御だって Pascal のような言語

で書けたなどという報告[45]もあるから簡単に誤解するむきが少なくないようで嘆かわしい．Pascalの処理系はPascalで書いてあるからあのように能率が悪いので，学生が実習でハイノの塔を走らせるのと同じ能率では困るのである．(もっとも能率は処理系の作製技術にも大きく左右される[46]．) この件についてMITのLiskovは筆者に「その言語の処理系がその言語で書けたからその言語はシステム記述言語だなんてとんでもない」といったし，チューリッヒのWirthは筆者に「Pascalは Pascalのシステム記述には使ったけれど，一般のシステム記述にはどうしても Pascalをだまさなければならない」といった．だが Pascalの処理系を Pascalで書いたのは処理系の解読も変更も容易だということであり，事実 Pascalが各地で勝手に改造される結果となったのである．筆者の研究室では Pascalで書いた Lispが走っているが[47]この Pascalも多少改造された，だまされた Pascalである．だが型有り言語は非常に有効であった．

Pascalに続いて Concurrent Pascalが出，またチューリッヒから Modula[48]や Modula-2[49]が発表されたけれども，これらが定着するかどうか．前二者はすでに過去のものである．特に Modulaはあまりにも PDP-11を意識しすぎの感なきにしもあらずだが，PL 360同様，システム記述言語はその辺が使い易いのかもしれない．

最近 Pascal-plusの論文[50]を読んだ．classのような考え方をとり入れた点は Concurrent Pascal流だが，普通はデータのイニシアライズしかないのに対し，これにはファイナライズのあるところが面白かった．システム記述言語としての関心は Ada[51]にもある．きっといろいろ書き易いのではないかと思うが結論はもう少し先にのばそう．

7. おわりに

そろそろファイナライズすべき頃となった．操作システムが新時代を迎えた，あるいは新しい時代が近いように感じていたので，それを裏づけそうな話題を操作システム側の努力の中からほんのわずかだが拾ってみたつもりである．いやそれは独断で，操作システムの現実はまだまだ暗黒時代だと判断されている読者があってもそれは一向に構わないが，進歩をつづけていることだけは認めていただけるかと思う．

うえに，操作システム側からの努力と書いたけれどその意味は，操作システムに対しては環境の方からも変化が起きつつあることを指摘したかったからである．つまりMulticsがあれほど大変だったのは複数の利用者のためのシステムを建設しようとしたからで，それは限られた資源を有効に利用するためには，複雑な制御を必要とするという原則を地でいったからである．単線区間で輸送力を増強するのに，幅輳した信号系統や神業の運転ダイヤで実現しているのに類似していた．だが複線化するのがもしも簡単ならば，誰も好んで複雑なシステムをもった単線方式は採用しないであろう．今や操作システムの直面している状況もそれに似ているといえよう．つまり計算機は共同利用する形態から個人で持つ時代へと変りつつあり，Triposは早くもそれに向けて設計された．こうなるとMulticsを泣かせた情報保護機構はあまり問題にならなくなる．最近同様の趣旨のもうひとつの操作システムが発表された．

Xerox の Pilot[52]である．これは構成からみると以前からその研究所で使われていた Altoに似ているが両者の関係はわからない．Pilotはシングルユーザ用のシステムであるだけでなく，操作システムがサポートする言語もただひとつ Mesaに固定してしまうという徹底ぶりで，これまでの操作システムが，そのサポートする言語の入出力要求でむやみとねじまげられていたことを思うと，これはまた大英断である．このような環境側の事情の好転で，操作システムの脱皮はますます早まるものと思われる．そして操作システムとは何であったのかが明快に解明される日が刻々と迫っていると考えられる．

参考文献

1) 高橋秀俊：ソフトウェア危機，情報処理，Vol. 10, No. 6, pp. 373-374 (1969).
2) Stoy, J. E. and Strachey, C.: OS 6—An Experimental Operating System for a Small Computer. Part 1: General Principles and Structure, Computer J., Vol. 15, No. 2, pp. 117-124 (1972).
3) Stoy, J. E. and Strachey, C.: OS 6—An Experimental Operating System for a Small Computer. Part 2: Input/Output and Filing System, Computer J., Vol. 15, No. 3, pp. 195-203 (1972).
4) Introduction to Multics, MAC TR-123, p. 208, Project MAC, M. I. T. (1974).
5) Saltzer, J. H.: Protection and the Control of Information Sharing in Multics, Comm. ACM, Vol. 17, No. 7, pp. 388-402 (1974).

6) Schroeder, M. D., Clark, D. D., Saltzer, J. H. and Wells, B. H.: Final Report of the Multics Kernel Design Project, MIT/LCS/TR-196, p. 111, Laboratory for Computer Science, M. I. T. (1977).
7) Schroeder, M. D., Clark, D. D. and Saltzer, J. H.: The Multics Kernel Design Project, Operating Systems Review, Vol. 11, No. 5, pp. 43-56 (1978).
8) Jensen, K. and Wirth, N.: Pascal User Manual and Report, LNCS 18, p. 170, Springer-Verlag (1974).
9) Richards, M. and Whitby-Strevens, C.: BCPL —The Language and its Compiler, p. 173, Cambridge University Press (1979).
10) Dijkstra, E. W.: The Structure of the THE Multiprogramming System, Comm. ACM, Vol. 11, No. 5, pp. 341-346 (1968).
11) Liskov, B. H.: The Design of the VENUS Operating System, Comm. ACM, Vol. 15, No. 3, pp. 144-149 (1972).
12) Strachey, C. and Stoy. J.: The Text of OS Pub, p. 131, Programming Research Group, Oxford University Computing Laboratory (1972).
13) Strachey, C. and Stoy, J.: The Text of OS Pub (Commentary), p. 155, Programming Research Group, Oxford University Computing Laboratory (1972).
14) Brinch Hansen, P.: The Solo Operating System: A Concurrent Pascal Program, Software-Practice and Experience, Vol. 6, No. 2, pp. 141-150 (1976).
15) Brinch Hansen, P.: The Solo Operating System: Job Interface, Software-Practice and Experience, Vol. 6, No. 2, pp. 151-164 (1976).
16) Brinch Hansen, P.: The Solo Operating System: Processes, Monitors and Classes, Software-Practice and Experience, Vol. 6, No. 2, pp. 165-200 (1976).
17) Brinch Hansen, P.: Disk Scheduling at Compile Time, Software-Practice and Experience, Vol. 6, No. 2, pp. 201-206 (1976).
18) 小川貴英, 浅井義幸: Solo Operating System の Implementation とその使用経験, 第18回プログラミング・シンポジウム報告集, プログラミング・シンポジウム委員会, pp. 4-11 (1977).
19) 斎藤康己: システム記述言語, 東京大学工学部計数工学科卒業論文 (1976).
20) Ritchie, D. M. and Thompson, K.: The Unix Time-sharing System, Comm. ACM, Vol. 17, No. 7, pp. 365-375 (1974).
21) Kernighan, B. W. and Cherry, L. L.: A System for Typesetting Mathematics, Comm. ACM, Vol. 18, No. 3, pp. 151-157 (1975).
22) 木村 泉: カーニハン氏とオンライン写植機, bit, Vol. 8, No. 5, pp. 40-46 (1976).
23) Kernighan, B. W. and Plauger, P. J.: The Elements of Programming Style, p. 147, Bell Telephone Laboratories (1974).
木村 泉訳: プログラム書法, p. 198, 共立出版 (1976).
24) Kernighan, B. W. and Plauger, P. J.: Software Tools, p. 338, Addison-Wesley (1976).
25) Dolotta, T. A. and Mashey, J. R.: An Introduction to the Programmer's Workbench, Proc. Second Int. Conf. on Software Engineering, pp. 164-168 (Oct. 13-15, 1976).
26) Ivie, E. L.: The Programmer's Workbench—A Machine for Software Development, Comm. ACM, Vol. 20, No. 10, pp. 746-753 (1977).
27) 石田晴久: ベル研究所の軽装 OS-UNIX, 情報処理, Vol. 18, No. 9, pp. 942-949 (1977).
28) 木村 泉: Software Tool とは何か? 情報処理, Vol. 20, No. 8, pp. 673-680 (1979).
29) Special Issue on Unix Time-Sharing System, BSTJ, Vol. 57, No. 6 Part 2, pp. 1897-2312 (1978).
30) Kernighan, B. W. and Mashey, J. R.: The Unix Programming Environment, IFIP WG 2.7 Bulletin, No. 4, pp. 71-84 (1978).
31) Machey, J. R.: PWB/UNIX Shell Tutorial, IFIP WG 2.7 Bulletin No. 4, pp. 85-92 (1978).
32) Kernighan, B. W. and Ritchie, D. M.: The C Programming Language, p. 228, Prentice Hall (1978).
33) Lycklama, H.: Unix on a Micro-processor, AFIPS Conf. Proc., Vol. 46, pp. 237-242 (1977).
34) Levinsky, J.: CHAOS—An Interactive Time-shared Operating System for the 8080, Dr. Dobb's Journal, Vol. 8, No. 1, pp. 6-13 (1979).
35) Lions, J.: Experiences with the Unix Time-Sharing Systems, Software-Practice and Experience, Vol. 9, No. 9, pp. 701-709 (1979).
36) Walker, B. J., Kemmerer, R. A. and Popek, G. J.: Specification and Verification of the UCLA Unix Security Kernel, Comm. ACM, Vol. 23, No. 2, pp. 118-131 (1980).
37) Cheriton, D. R., Malcolm, M. A., Melen, L. S. and Sager, G. R.: Thoth, a Portable Real-Time Operating System, Comm. ACM, Vol. 22, No. 2, pp. 105-115 (1979).
38) Richards, M., Aylward, A. R., Bond, P., Evans, R. D. and Knight, B. J.: TRIPOS-A Portable Operating System for Mini-Computers, Software-Practice and Experience, Vol. 9, No. 7, pp. 513-526 (1979).
39) Frank, G. R. and Theaker, C. J.: The Design

of the MUSS Operating System, Software-Practice and Experience, Vol. 9, No. 8, pp. 599-620 (1979).
40) Brinch Hansen, P.: Operating System Principles, p. 366, Prentice Hall (1973).
田中，真子，有沢訳：オペレーティングシステムの原理，近代科学社 (1976).
41) Lauer, H. C. and Needham, R. M.: On the Duality of Operating System Structures, Operating Systems Review, Vol. 13, No. 2, pp. 3-19 (1979).
42) Baker Jr, H. G.: Shallow Binding in Lisp 1.5, Comm. ACM, Vol. 21, No. 7, pp. 565-569 (1978).
43) Corbató, F. J.: PL/I as a Tool for System Programming, Datamation, Vol. 15, No. 5, pp. 68-76 (1969).
44) 木下 恂：Pascal はシステム記述言語に適しているか？，情報処理，Vol. 21, No. 2, pp. 180-182 (1980).
45) Hoare, C. A. R.: A Structured Paging System, Computer J., Vol. 16, No. 3, pp. 209-215 (1973).
46) 白浜律雄，前野年紀：字句解析部の高速化について，第 20 回プログラミング・シンポジウム報告集，プログラミング・シンポジウム委員会，pp. 8-18 (1979).
47) 近山 隆：Pascal による Lisp 処理系の作成，第 20 回プログラミング・シンポジウム報告集，プログラミング・シンポジウム委員会，pp. 118-125 (1979).
48) Wirth, N.: Modula: A Language for Modular Multiprogramming, Software-Practice and Experience, Vol. 7, No. 1, pp. 3-35 (1977).
49) Wirth, N.: Modula-2, p. 35, Institut für Informatik, ETH (1978).
50) Welsh, J. and Bustard, D. W.: Pascal-Plus —Another Language for Modular Multiprogramming, Software-Practice and Experience, Vol. 9, No. 11, pp. 947-957 (1979).
51) Department of Defence: Preliminary ADA Reference Manual, Sigplan Notices, Vol. 14, No. 6 (1979).
52) Redell, D. D., et al.: Pilot: An Operating System for a Personal Computer, Comm. ACM, Vol. 23, No. 2, pp. 81-92 (1980).

(昭和 55 年 4 月 2 日受付)