オペレーティングシステム開発における「環境問題」

電気通信大学・電子情報学科　　多田　好克
Tada Yoshikatsu

概　要

ソフトウェアの生産性はソフトウェアを作成する環境に大きく左右される。オペレーティングシステムの提供する基本機能やプログラム開発用道具が貧弱では、良いソフトウェアを効率良く開発することはできない。開発するソフトウェアがオペレーティングシステムの場合には、事態はさらに深刻になる。本論文では、このような現実を踏まえ、オペレーティングシステム開発効率を左右する様々な要因を調査、分類する。さらに、いくつかの開発環境を例にとってそれらを比較、検討し、具体例を示しながら「究極のオペレーティングシステム開発環境」とはどのようなものかを考える。

1．　はじめに

我々の研究室では、自前のプログラミング言語による自前のＯＳ開発を目指して、その研究環境、開発環境の整備を行っている[1]。そこで本論文では、我々が行っているＯＳ開発計画の背景を述べ、ＯＳ開発のために準備している環境の得失を議論する。

まず次章ではＯＳ開発の意義について、また、続く３章ではＯＳ開発環境を整えることの意義について、それぞれ簡単に論じる。４章ではＯＳ開発環境を４つの構成要素に分類し、各要素に要求される性質や機能を議論する。５章はＯＳの移植や拡張を行った経験に基づく章で、その時に使用した開発環境の利点、欠点を議論する。続く６章では、我々が構築を試みているいくつかのＯＳ開発環境を紹介し、それぞれの環境を比較、検討する。また、７章では我々の構築しているＯＳ開発環境の現状を紹介し、８章で今後の問題点をまとめる。

Environments for Operating System Development --- Their Problems and Advantages.
Yoshikatsu Tada.
The Univ. of Electro-Communications.

2．　何故ＯＳ開発か？

ソフトウェアの生産性は、ＯＳの提供する基本機能や、利用可能なプログラム開発用道具の良否によって大きく左右される。また、文字数に上限のあるファイル名や半二重の通信方式等、制限のあるＯＳの上に開発されたプログラムは、すべてその制限を受け継ぎ、保守性や使い心地に問題を生じることになる。つまり、良いソフトウェアを作成するためには、良いＯＳの存在が必要条件であると考えられる。

現在、ソフトウェア作成者の間で高い評価を受けているＯＳにUnix[2]がある。Unixには、①構造を持たないファイル、②階層構造を持ったファイルシステム、③抽象化された入出力装置、④一般化された入出力操作、⑤明確なプロセスの概念、⑥ソースコードの存在、といった特長があり、また、その上には多数

のプログラム作成用道具が開発されている。

しかし、このUnixにも問題がないわけではない。我々は約10年の間、研究用、実験用のＯＳとしてUnixを使用してきたが、その間に、以下に示すような問題点が明らかになってきた。

i) ライセンスの問題

Unixのソースコードを参照、変更するためにはＡＴ＆Ｔとライセンス契約を結ぶ必要がある。ＯＳの実現法に関する議論等はこの契約によって制限を受ける。

ii) 地盤崩壊の問題

ＯＳの研究をするため、または、ＯＳをより使い易くするためには、様々な改良が必要である。たとえば、ＯＳの日本語化は日本で使用するＯＳには不可欠の問題と考えられる。しかし、Unixに改良を加えた場合、Unix自体のバージョンアップに追随しにくくなる。改良した古いUnixを利用するか、新しいバージョンのUnixに乗り換えるかは、Unixを利用している場合、常に直面する大問題である［図１］。

iii) 概念の風化問題

Unixの基本概念は約20年前に作られたものである。したがって、それ以降に考えられたウィンドウシステムや軽いプロセス等の概念をエレガントに実現するにはＯＳ自体の大幅な改良を必要とする。同様に、ビットマップディスプレイやネットワークインタフェースのドライバを記述する際にも多数の問題を解決しなければならない。

iv) Unixの肥満問題

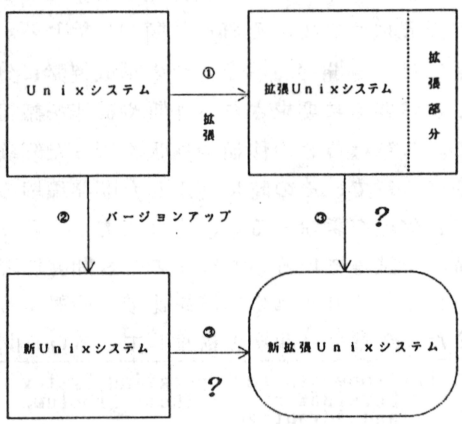
SystemＶ、4.3BSD等の最近のUnixでは様々な機能拡張により、ＯＳ自体が巨大化している。そのため、ＯＳの研究、実験には適さなくなった[3,4]。

以上のような問題点を克服するためには、仕様変更や改良の容易なＯＳを、基本部分から再構築するのが良い。Unixが一般的になった現在でも、ハードウェア事情や新しい概念を考慮に入れた新しいＯＳの開発は有意義であると考えられる。また、現在、分散ＯＳのカーネルやマルチプロセッサシステムの核にUnixを流用したシステムが多数発表されている。しかし、専用の単一CPU用ＯＳを開発し、それを核にした分散ＯＳやマルチプロセッサシステムを作成した方が実行効率や保守性等の面で優れていると考えられる。

以上のような理由により、研究、実験の基礎となるＯＳを新たに作成することは、研究としてのみならず、実用的な観点からも重要であると思われる。

３．　何故ＯＳ開発環境か？

前章で議論したような、新しいＯＳを開発する場合には、その開発環境が重要である。ただ単に、実用的なＯＳを作成するだけであれば、ある程度の開発環境を用意した後、直ちにＯＳの作成を始めた方が、目的達成に要する労力は少ないかもしれない。しかし、２章で論じたように、ＯＳに関する研究を長期に渡って行う場合には、開発環境の良否が研究の効率を左右する。



［図１］　Ｕｎｉｘの拡張とバージョンアップの関係

これは、機械製品と工作機械との関係にも似ている。様々な製品を作成し、それらの良否を検討する場合には、良い工作機械の存在、すなわち、整備された開発環境の存在が必須である。また、実際、良い開発環境によって研究、実験に要する労力は削減できると考えられる（［図２］：最終ページ参照）。

４．　ＯＳ開発環境

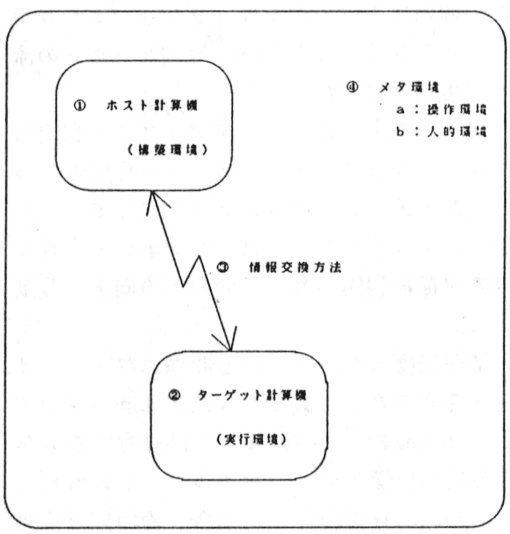
我々は、ＯＳ開発環境を以下に示す４つの構成要素からなると考えた［図３］。

①　ホスト計算機（構築環境）
②　ターゲット計算機（実行環境）
③　情報交換方法
④　メタ環境（④a:操作環境、④b:人的環境）

なお、仮想計算機[5]を使ったＯＳ開発環境では、①と②とに同一計算機を使用し、これに伴って③が簡略化される。しかし、ワークステーション等のハードウェアが安価に入手できる現状を考慮し、本論文では仮想計算機によるＯＳ開発環境は考えない。

以下、各構成要素について簡単に説明する。

まず、①のホスト計算機は開発ＯＳを実際にエディット、コンパイルする環境である。ＯＳ記述用の言語処理系も本計算機上に構築される。この構築環境には、プログラミング用環境とプログラミング用道具が必須である。また、ターゲット計算機の動作実験用に適当な言語の（クロス）処理系、ＯＳ記述言語作成用に適当な言語処理系がそれぞれ必要である。



［図３］　ＯＳ開発環境の構成要素概念図

②のターゲット計算機は開発ＯＳを実際に実行する環境である。この実行環境には、アーキテクチャが一般的であること、アーキテクチャ等の資料が完備していること、が要求される。また、ＯＳを仮定しないモニタ、デバッガの存在も、望ましいターゲット計算機の条件である。さらに、開発環境の良否には直接関係しないが、多数の同一機種が存在していることも重要である。

③の情報交換方法では、まず、その転送速度が問題になる。また、情報の転送に人手を介さないというのも重要である。フロッピーディスクを抜き差しして情報を転送するよりは、専用の回線を通して自動的に情報を転送できる方が環境としては望ましい。情報交換方法に対する３つ目の要求は、ターゲット計算機からホスト計算機へも情報が転送できることである。これにより、開発中のＯＳのイメージをホスト計算機上で解析すること等が可能となる。

④のメタ環境は、操作環境（④a）と人的環境（④b）に分けて考える。たとえば、ターゲット計算機のコンソール設置場所のクーラーが効き過ぎているというのは操作環境の問題、電話によって仕事を中断されるというのは人的環境の問題と考える。メタ環境は構築環境、実行環境の存在場所に依存する。ところで、「開発環境を夜中まで使えるか？」といった問題は、一見、操作環境の問題のようにも思えるし、また、建物の管理という人的環境の問題と考えることもできる。④aと④bとの違いを明確にするのは難しいが、以下では、で

きる限り両者を区別し、開発環境問題の根本原因を究明する。

なお、続く５章、６章では、上述の分類に従ってＯＳ開発環境の問題点を議論する。これらの章では、①～④の記号によってＯＳ開発環境の各構成要素を参照する。

５．　既存の環境とその問題点

筆者は、1984年にsUNixの実現[6,7]、1987年にt-process（ある種の軽いプロセス）の実現[8]、と２度に渡ってＯＳの移植、改良を経験した。この章では、これらに使用したＯＳ開発環境を紹介し、その問題点を議論する。

5.1.　sUNix開発環境

まず、sUNix開発環境の構成要素を以下に示す。

① VAX-11/780, Unix 4.1BSD.
② Sun-1. 8KbyteのROMモニタ.
③ 9600baud RS232C回線.
④ 東京大学大型計算機センター(構築環境).
　東京大学工学部情報工学専門課程
　和田研究室(実行環境).

構築環境は、使い易いという定評のあるバークレー版のUnixと、当時Unixの動く計算機としては高速のVAX-11/780である。vi, makeを初めとする様々なプログラミング用道具があり、安定したＣ言語処理系とライブラリやユーティリティが用意されていた。ターゲット計算機用のクロス言語処理系にはMITで作成された68000用Ｃコンパイラを利用した。

実行環境のSun-1は、初代のSunワークステーションである。当時、Sunマイクロシステムズ社ではSunに、まだ、Unixを移植中であった。そのため、このSun-1には入出力装置を初めとする各種ハードウェアのマニュアルが付属していた。また、この計算機には8KbyteのROMモニタが用意されており、RS232C回線によるデータの送受や簡単なデバッガ等の機能が提供されていた。

情報の交換には、9600baudのRS232C回線を利用した。データの送受にはモトローラのＳフォーマットを使用したため、転送効率は45％程度と推定される。

メタ環境は、構築環境がセンター、実行環境が和田研と、２ヶ所に分断された。両環境間は前述のRS232C回線によって接続され、ターゲット計算機をホスト計算機の端末として利用することも可能であった。また、両環境間の移動には徒歩で約３分を要した。

本開発環境には、問題点が多数存在した。以下は、本開発環境の問題点、および、特長である。

まず、構築環境では不安定なクロスコンパイラに泣かされた。コンパイラによる虫に気を配りながら、ターゲット計算機の動作実験を行うのは至難の業であった。

実行環境では、ビットマップディスプレイという特殊なハードウェアが存在したため、その部分のプログラミングに余分な時間を要した。ＯＳ開発の目的にもよるが、一般に、開発当初は特殊なハードウェアの使用を避けることが望ましい。また、モニタプログラムの仕様が不明確であったため、逆アセンブルをかけて、一部その内容を解読した。なお、この解読により、メモリリフレッシュやコンソールドライバ等の割込み処理ルーチンの流用が可能となった。

実行環境上には、ハードディスクを利用したローカルなファイルシステムが作成され、メモリイメージの保存、回復が可能であった。このファイルシステムは、開発中のカーネル等の保存に利用され、開発効率の向上に貢献した。

情報交換において、最も問題になったのは、転送速度の遅さであった。転送速度の実測値は430byte/秒である。ホスト計算機の負荷が比較的軽い深夜に、ファイルシステムのイメージ500Kbyteを転送した場合、約20分の時間を必要とした。また、カーネルの転送にも３分を要し、虫取りの効率を下げる原因となっ

た。

メタ環境に関しては、所属研究室内にターゲット計算機があったため、良好な操作環境を構築できた。机が狭い、端末周辺が（学生同士の会話で）騒がしい等の問題はあったが、大学院生という立場を考えると、満足のいく環境であったと考えられる。他方、人的環境に関しては、事務の管理体質がＯＳ開発の妨げになった。19時以降は研究室への出入りにＩＤカードを必要としたし、21時以降は研究室のある建物からの退出を要求された。最終的には教官用のＩＤカードを教授より借り受け、徹夜可能な環境となったが、このＩＤカードなしにＯＳを開発することは不可能であったと思われる。大型計算機センターの管理も同様であり、18時以降、磁気テープの読み書きができなくなるのは大問題であった。

メタ環境、特に、人的環境は、管理的な事務と、友好的な研究室スタッフとの力関係に大きく左右されると考えられる。

ＯＳ開発に関する議論ができる研究者や虫取りに協力してくれる研究者が和田研内に多数いたことも、ＯＳ開発環境の問題としては見逃すことができない。同じような問題意識を持った研究者の存在は、人的環境の良否を大きく左右する。

5.2. t-process開発環境

t-process開発環境の構成要素は以下のとおりである。

① VAX-11/750, Unix 9th Ed.
② VAX-11/750. コンソールサブシステム.
③ ハードディスク.
④ AT&T Bell Laboratories.

構築環境と実行環境は同一ハードウェアを使用した。構築環境のＯＳはBell研内部で使用しているUnixで、バークレー版のUnix同様使い易い環境であった。ホスト計算機の処理速度は比較的遅いと考えられるが、利用者が通常はひとりしかいないこと、インテリジェント端末がウィンドウ処理を実行すること、により問題にはならなかった。

実行環境はVAX-11/750である。これには、アーキテクチャ、入出力装置等に関するマニュアルが完備していた。また、実際に構築環境用のUnixがこの計算機上で稼働しているので、そのソースコードも重要な情報源であった。なお、開発したＯＳの起動にはコンソールサブシステムを利用した。

情報の交換には、ハードディスクを利用した。ファイルシステム等の形式は同一なので、構築環境でファイルを作成し、それを実行環境で読むという方法を採用した。

構築、実行用の計算機は計算機室に置かれていた。この部屋は冷房されており、劣悪な操作環境であった。しかし、VAX-11/750のコンソールを研究室の端末のウィンドウに設定することができ、通常は端末のウィンドウを通して実行環境を制御可能であった。

本開発環境の特長、および、問題点は以下のとおりである。

まず、構築環境の作成に使用したＣコンパイラを、そのまま開発用コンパイラとして利用できたため、前節で論じたようなコンパイラの虫による問題がまったく生じなかった。

実行環境には、多数のマニュアルと実際に稼働しているＯＳのソースコードがあったので、開発ＯＳのプログラミングは比較的容易であった。ただし、構築環境とファイルシステムを共有していたため、開発ＯＳの暴走によってファイルシステムを破壊するという危険と常に直面していた。

情報交換は、ハードディスクを利用したため高速に行えた。しかし、構築環境にて情報を保存した後、構築環境の停止、実行環境の起動の手順を踏むため、開発環境全体の効率は悪かった。また、試験的な実行が終了した後にも構築環境の再起動が必要であった。

開発ＯＳ実行時のイメージ等は、ハードディスクを使って構築環境に移すことができる。この機能によって、開発ＯＳの虫取り効率が

向上した。

操作環境は良好であった。特に、ターゲット計算機のコンソールを通常の端末回線に接続できたため、研究室のみならず自宅から電話回線を通して開発ＯＳを走らせることも可能であった。また、研究室へは終日出入りでき、思い付いたアイデアを直ちに実験することが可能であった。研究者には、基本的にひとり一部屋が割り当てられ、空調等も快適であった。たとえば、部屋の蛍光灯が古くなった場合に、電話連絡さえすれば３時間以内に担当者が交換にくるといった環境は、特筆に値する。

人的環境も良好であった。研究者間の連絡は電子メイルを使って行われる。電話による連絡は思考を妨げるというのがその理由である。また、複数台の端末を準備した大部屋があり、議論をする環境も整っていた。

６．　４つのＯＳ開発環境例

この章では、４章、５章で論じたＯＳ開発環境の問題点を考慮した上で、現在、我々が構築を進めている４つのＯＳ開発環境を紹介し、これらの開発環境の得失を比較、検討する。なお、この章では、各開発環境の特長に注目し、以下に示す題を付けた小節に分け、それぞれについて議論する。

a) 勝手知ったるPC-98.

b) 5"2HDは不滅です.

c) マニュアルのある拡張CPUボード.

d) ネットワークでbooting.

なお、b),c),d)の環境については、ターゲット計算機を製造している会社より社内資料の提供を受けているため、会社名、計算機名を明言していない。また、以下の議論において、④のメタ環境は４つの開発環境すべてに共通と考えられるので、これについてはこの章の終わりで一括して議論する。

6.1.　勝手知ったるPC-98

この環境では、CPUに80386を搭載したNECのマイコン、PC-9801RA等をターゲット計算機に採用する。ターゲット計算機のCPUに80386を選んだ理由等は後述する。

この場合、構築環境には、

①a PC-9801, MS-DOS

①b PC-9801, Unix

①c CPUが80386以外のワークステーション

①d Sun-386i

の４種類が考えられる。①aの環境は、比較的安価に用意できるが、プログラミング環境が貧弱なこと、ターゲット計算機を試験するためのＣコンパイラが不安定なこと、が問題となる。①bの環境では、Unixに付随のＣコンパイラをターゲット計算機の試験用に利用できる。このコンパイラは、Unix移植にも使用されたはずなので、コンパイラに関する不安は減少する。しかし、この①bの環境では、5.2節で論じたように構築環境の停止と起動に要する時間が開発効率を低下させる。①cの環境では、gnuのＣコンパイラを利用することになるが、その安定性は未知数である。また、情報交換方法をどうするかも問題になる。現在、比較的安価に手に入るワークステーションで、そのCPUに80386を採用しているのはSun-386iである（①d）。この環境の問題点は情報交換の難しさにある。次節で論じる方法と同様にフロッピーディスクを利用することも考えられるが、そのための作業量は馬鹿にならない。

PC-9801をターゲット計算機に採用することの利点は少なくない。まず、ターゲット計算機としては安価であり、入手も容易である。NECから提供される資料の類は貧弱であるが、多数の解説本が出版されており、入出力装置等とのインタフェースも簡単に知ることができる。MMUの仕組みも80386のマニュアルに明確に定義されている。また、80386のセグメントレジスタは軽いプロセスの実現に有用であると考えられ、我々が開発を目指しているＯＳの基本部分の実現に都合がよい。

この実行環境には問題点も多い。PC-9801の

入出力装置は初期バージョンのものをそのまま流用しており、現在では時代遅れになりつつある。また、入出力装置の拡張性も乏しい。入出力装置のインタフェースに関する情報は解説本が頼りのため、たとえばイーサーネットボードのような特殊な装置の情報は手に入れることができない。

情報交換方法は、構築環境①a～①dに依存する。①aの効率は良いが、その他の場合には工夫が必要になる。①bでは、ホスト計算機とターゲット計算機の２台の計算機を用意することにより、Unixシステムの起動、停止時間の問題を解決できる。ただし、ホスト計算機のフロッピーディスクをターゲット計算機で読めるかどうかは未知数である。①c、①dでは、イーサーネットボードを追加し、PCNFSを利用することが考えられる。しかし、前述したように、このイーサーネットボードを開発ＯＳでも使用するためには資料が不足しているようである。

6.2.　5"2HDは不滅です

この節で論じる開発環境は、

①　我々の学科にあるワークステーション群
②　Ａ社のミニコンａ(CPU:68000)
③　PCNFSと5"2HDフロッピーディスク

の各要素より構成される。以下、それぞれについて得失を議論する。

構築環境には、Sun-3, NEWS, Luna等のワークステーションを使用する。これらのホスト計算機は、68000系のCPUを持ちバークレー版のUnixをＯＳに採用している。また、我々の学科のLAN(uecrnet)によって、互いに接続されている。この構築環境で問題になるのは、ターゲット計算機の動作実験に使用するコンパイラである。当初、ホスト計算機のコンパイラを流用すれば良いと考えていたのだが、これらは68020/68030用のオブジェクトを出力するため利用できなかった。代替コンパイラとして、gnuのＣコンパイラとsUNix実現時に利用したMITのＣコンパイラとを考えた。結局、sUNixの移植を終えて動作が安定していること、コンパイラが小さいこと、実現が簡単であること、の理由により、現在はsUNix開発時のものを使用している。

実行環境は、「一昔前の個人用Unix計算機」である。CPUに68000を採用し、メモリ管理機構、ハードディスク、ビットマップディスプレイ等が備わっている。SystemＶのUnixを起動することも可能であるが、上述のホスト計算機に比べれば、速度、使い心地共に劣り、利用者もほとんどいないのが、この計算機の特徴である。

我々の研究室では、Ａ社にお願いして回路図や入出力装置とそのアドレスに関する資料を入手した。また、モニタROM、IPL、booterの各プログラムも計算機処理可能な形式で手に入れた。これらのソフトウェアはUnixのライセンス外なので、ターゲット計算機製造会社の方針によっては入手可能な場合がある。

この実行環境は、一般的なアーキテクチャを持ち、その資料も整っている。入出力装置には汎用のLSIが使われ、そのマニュアルも容易に入手できる。

この実行環境には、PC-9801の場合と同様、ターゲット計算機のアーキテクチャが古いという問題がある。また、この機種の製造は既に打ち切られており、今後の保守には不安が残る。

入手したアーキテクチャの資料は社内向けのため、情報の過不足が目につく。足らない情報に関しては、モニタROM等のソースコードで補うことが可能であるが、開発効率の低下は免れない。

情報交換は２段階の手順を踏む。まず、構築環境で作成したファイルを、PCNFSを使ってPC-9801の5"2HDフロッピーディスクに書き出す。ファイルシステムの形式はMS-DOSのものをそのまま利用する。次に、そのフロッピーディスクをターゲット計算機Ａに挿入し、新たに作成したbooterによって読込み、実行す

る。（このbooterの詳細は７章で改めて紹介する。）

この方法では、フロッピーディスクの抜き差しの速度が、情報交換に要する時間を左右する。現在、筆者の使用している環境では、数秒程度で情報交換が可能である。

上述の情報交換法を実現するためには、ターゲット計算機上にMS-DOS形式のフロッピーディスク用ルーチンを作成しなければならない。我々が作成したものは、約800行のＣと約50行のアセンブラより構成される。このルーチンはMS-DOS形式で格納されたファイル名を列挙し、キーボードより入力された名前のファイルを実行する。キーボードや画面の入出力ルーチンは本来のbooterルーチンを借用したため、比較的少ない行数でこのような機能を実現できたと思われる。

本情報交換法では、フロッピーディスクに格納した複数のプログラムを、ファイル名を指定して選択実行することができる。そのため、sUNix開発時のローカルなファイルシステム同様、開発効率を高めることが可能である。

6.3. マニュアルのある拡張CPUボード

拡張CPUボードには、通常、アーキテクチャ等の資料が提供される。この資料は良く整備されており、ＯＳ開発には好都合な場合が多い。我々が考えている３つ目の開発環境は、Ｂ社の拡張CPUボードｂを実行環境に使用する。本開発環境の構成要素は、

① Sun-3/160（拡張バス付き）
② 拡張CPUボード（マイクロプログラム可）
③ VMEバスを使ったメモリ間転送

である。

構築環境では、４章で論じた機能の他に、ターゲット計算機の入出力をも代行しなければならない。独立した計算機上で作動するＯＳの開発を目指す場合には、これらの代行入出力操作を実際のハードウェア機構に準じて作成しておくことが望ましい。既存のハードウェアの割込み機構等を代行するソフトウェアの開発は、相当量のコーディングを必要とすると考えられる。

拡張CPUボードの動作を確認するためには、構築環境上にクロスコンパイラを作成しなければならない。拡張CPUボードを製造している会社より言語処理系が提供されることもあるが、比較的高価で不安定な場合も多い。

前述したように、本開発環境は、ターゲット計算機のアーキテクチャに関する資料を容易に入手できるという特長を持っている。また、入出力装置とのインタフェースは構築環境のプログラムによって規定されるため、ＯＳ開発を効率的に行える。

他方、他の実行環境に比べると入出力操作が遅いという欠点がある。構築環境の入出力代行プログラムに手を入れれば、開発したＯＳの実行効率は評価可能である。したがって、研究自体に支障をきたすことは少ないと考えられるが、デモの実行速度が遅くなるのは必至であり、開発したＯＳを低く評価される恐れがある。

情報交換は高速である。構築環境上の情報を必要な時に実行環境に転送できるため、効率の良いＯＳ開発環境が実現可能である。

6.4. ネットワークでbooting

近年のLANの発達により、ネットワーク上のサーバよりディスクレスワークステーションへＯＳを転送し、起動できるようになった。この機能を利用したＯＳ開発環境は、

① ディスクサーバ
② ディスクレス構成のディスク付き計算機
③ ネットワーク

の各構成要素より構築される。

構築環境には、6.2節に示したワークステーションを使うことができる。booterやコンパイラの整合性を考えるならば、ターゲット計算機と同じ会社のワークステーションを利用するのが望ましい。

実行環境にはＣ社のディスク付きワークステーションｃを使用する。近年の低価格ワー

クステーションの中には、I/Oプロセッサを搭載したものや専用の入出力LSIを搭載したものが多い。これらの部分のインタフェースが明かでないものは、本開発環境のターゲット計算機には不向きである。ワークステーションｃは数チップの専用入出力LSIを使用しているだけであり、他の計算機に比べるとそのアーキテクチャは把握し易い。また、6.2節で論じた実行環境同様、Ｃ社よりbooter等のソースコードが入手できる予定である。

本開発環境では、構築環境上の任意のファイルを実行環境上で走らせることができる。情報交換にはLANを利用できるため、開発効率は高いと考えられる。しかし、開発したＯＳのディスク入出力をサーバに頼るためには、新たにネットワークドライバとTCP/IP等の上位通信層を作成しなければならない。このドライバはいずれ必要になるものではあるが、ＯＳ開発の初期に作成する必要はない。本開発環境ではターゲット計算機にディスクを用意し、取り敢えずネットワークドライバの作成を後回しにすることを考えている。

6.5.　メタ環境に関する考察

この章で議論した４つのＯＳ開発環境は、必然的に電気通信大学のメタ環境下に構築されることになる。この節では、このメタ環境の問題点を一括して議論する。

我々の所属する電子情報学科では、現在、主要計算機の終日運転を実施している。また、構築環境、実行環境共に研究室内にあり、部屋の鍵さえあれば自由に操作できる。研究室から徒歩数分には24時間営業のコンビニエンスストアもあり、操作環境は良好であると考えられる。

一方、人的環境は劣悪であると言わざるを得ない。まず、施設の改善のために、今年度に入って４回停電があった。これらは、すべて日曜日ではあったが、計算機システムの停止、起動にはそれぞれ数時間を要した。さらに、契約電力量を越えたという理由で、今年度に入って２度、突然停電するという事態を経験した。

事務からの連絡は、各部屋に設けられたスピーカを通して行われる。しかし、電話の使用を控えるBell研の環境を考えると、これもＯＳ開発の効率を下げる一因である。その他、学内便の処理法等、より良いＯＳ開発環境を構築するためには、事務処理の効率改善が必要であると考えられる。

7　ＯＳ開発環境の現状

我々の研究室では、６章の冒頭で述べた、a)～d)の４つのＯＳ開発環境をb),d),c),a)の優先順位に従って構築中である。この章では、比較的構築の進んでいるb)の環境の現状を紹介し、その問題点を議論する。

ＯＳ開発用のbooter（z-booter）は、ターゲット計算機であるＡ社のミニコンａ上で作成した。これは、Ａ社から入手したbooterのプログラムの一部を流用したため、および、モニタROM内のIPLルーチンが想定しているオブジェクト形式にz-booterの形式を合わせる必要があったため、である。前述したように、Ａ社のbooterに約850行のプログラムを追加してz-booterは作成された。

z-booterはフロッピーディスク上のファイルを読み込み、そのイメージを実行する。ファイルは構築環境上のＣクロスコンパイラ、および、新たに作った開発用道具を使って作成される。ファイルには、ヘッダ、オブジェクトイメージ、データイメージが順に格納されている（［図４］：次ページ参照）。なお、オブジェクトイメージは直接実行されるプログラムのイメージ、データイメージはプログラム実行時のファイルやファイルシステムのイメージとして使用することを想定している。

z-booterはメモリ管理機構を初期化した後、オブジェクトイメージ、データイメージを主記憶上に読み込む。また、オブジェクトイメージの実行に備えてカーネルスタックポイン

タを設定する。［図５］にはオブジェクトイメージ実行直前のカーネル仮想空間の配置を示した。

z-booterはスタック上に４つの実引数を設定し、オブジェクトイメージ内の関数mainを起動する。したがって、mainの一般形は、

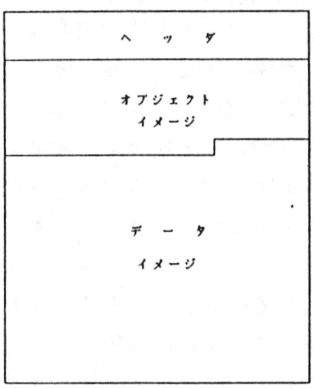
```
void main(argp, envp, page, free)
        char    *argp, *envp;
        int     page, free;
```

となる。引数argpは、実行ファイル名を指定した時のコマンド行へのポインタ、envpはファイルに格納されていたデータイメージへのポインタである。また、pageとfreeはそれぞれページマップ、実メモリの使用可能領域の境界を表す。

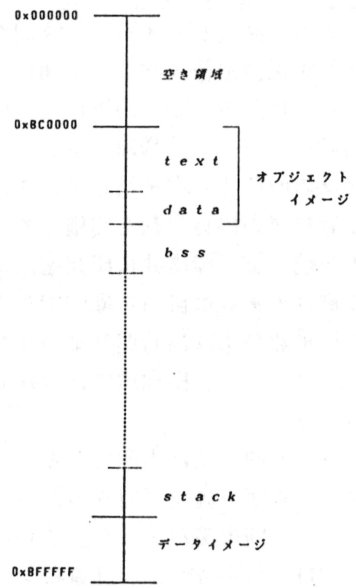
z-booterはＡ社より入手した資料、および、ソースコードを参照して作成した。また、フロッピーディスクドライバの部分は、MS-DOSのファイルシステムに関する本、および、制御ＩＣのマニュアル等をも参照した。これらの資料は比較的簡単に手に入れることができた。しかし、個々の情報から統合的なプログラムを作成するのは容易ではなかった。ターゲット計算機のアーキテクチャ設計の意図を記述した資料が望まれる。

z-booterを使用した場合の開発手順を、以下に順を追って説明する。

i) 構築計算機上のクロスコンパイラを利用して、オブジェクトイメージを作成する。
ii) 構築計算機上の開発道具を利用し、i)のファイルにデータイメージの追加を行う。最終的に、［図４］の構造を持ったファイルができあがる。
iii) PCNFSを利用して、PC-9801上のフロッピーディスクに、ii)のファイルイメージを複写する。
iv) iii)のフロッピーディスクをミニコンaに挿入する。
v) z-booterが、MS-DOSのファイル名を画面に表示するので、その中のひとつをキーボードから入力する。（ファイル名以降の文字列は、実行するプログラムの関数mainに引数として渡される。）
vi) オブジェクトイメージの実行が始まる。関数mainの実行が終わると、z-booterに制御が返ってくる。



［図４］　ｂｏｏｔファイルの構造



［図５］　ｂｏｏｔ直後の仮想空間の配置

vii) z-booterが、MS-DOSのファイル名を画面に再び表示するので、必要ならばv)からの操作を繰り返す。

現在は、構築環境上のCクロスコンパイラを利用して、ターゲット計算機の動作実験を行っている。特に、画面、キーボード、ハードディスク等の入出力装置のドライバを記述し、その動作を確認している。また、これと並行して、ＯＳ記述用の言語を設計し、構築環境上にそのトランスレータを実現している。

8．　おわりに

本論文では、現在、我々の研究室で構築を目指している４種類のＯＳ開発環境を紹介し、その得失を議論した。また、特に構築の進んでいる環境について、その現状を紹介した。

我々は、2,3章で論じた問題意識に基づいてＯＳ開発環境の設計を行ってきた。最近、これらの問題以外に、ターゲット計算機のCPUはRISCかCISCかという問題にも直面している。筆者自身はCISCの方が良いと思っているのだが、実際には議論の別れるところであろう。

良いＯＳ開発環境を構築するためには、4章で論じた様々な条件を満たさなければならない。特に、

① 構築環境はBSD系のUnixを使う
② 構築環境と実行環境に同一CPUを使用する
③ 資料の豊富なターゲット計算機を使う
④ 情報交換を高速にする
⑤ 操作性、居住性に気を配る
⑥ 他の研究者と頻繁に議論する
⑦ 事務との戦いに勝つ

の各条件が重要であるということが、我々の経験より得られた結論である。

［参考文献］

[1] 多田好克：”オペレーティングシステム開発環境について”，電気通信大学電子情報学科情報計測学講座輪講資料，Sep. 14 (1989).

[2] Ritchie, D. M. and Thompson, K.: "The Unix Time-Sharing System," Communications of the ACM, Vol. 17, No. 7, pp.365-375 (1974).

[3] Tanenbaum, A. S.: "A Unix Clone with Source Code for Operating Systems Courses," ACM Operating Systems Review, Vol. 21, No. 1, pp.20-29 (1987).

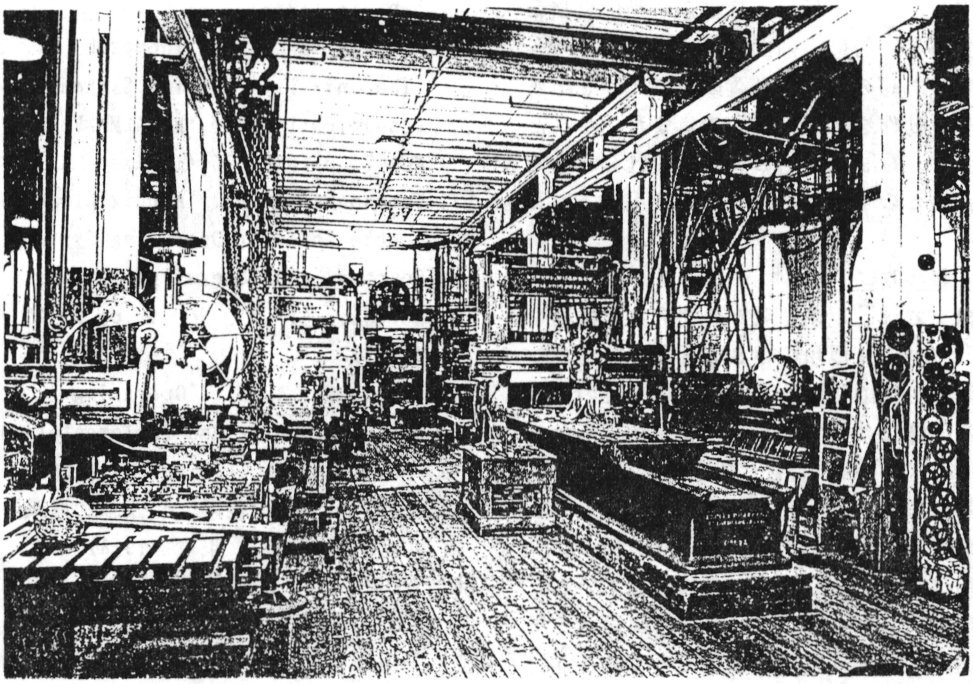
[4] Tanenbaum, A. S.: "Operating Systems: Design and Implementation," Prentice-Hall, New Jersey (1987).

[5] 山谷正己，秋山義博：”仮想計算機”，共立出版（株），東京 (1978).

[6] 多田好克：”Unix移植日記”，ｂｉｔ，Vol. 16, No. 13, pp.90-94 (1984).

[7] 多田好克：”Unixのワークステーションへの移植性について”，情報処理学会論文誌，Vol. 26, No. 6, pp.1033-1040 (1985).

[8] Tada, Y.: "Experimental Light-Weight Processes on 9th Edition Unix," AT&T Bell Labs. Tech. Memo. (1988).



［図２］　エジソンの研究所にある工作機械群

New Jersey州West Orangeにある"Edison National Historic Site"の工作室の風景。これらの工作機械の中には、エジソンに限り使用可というものも散見される。また、左手奥にはエジソン専用のエレベータが設置されており、エジソンの居る階には、常に、そのエレベータが待機していた。