第 17 部

オペレーティングシステム

第 1 章

はじめに

WIDEプロジェクトは1990年からカリフォルニア大学バークレー校の Computer Systems Research Group(CSRG) と共同研究を行っている。CSRGによるNET/2(Network Release 2) のリリースは、USLとの訴訟問題が発生し配布が一時中断されていたものの 1994 年の 2 月に解決、最終的に 4.4BSD-Lite として登場している。これによって我々は良質なオペレーティングシステムのソースコードを再び自由に入手することが可能になった。本論文はまず現在入手可能な 4.4BSD-Lite ベースのオペレーティングシステムを取り上げ、その特徴などについて述べる。

一方で我々を取り巻く計算機環境は大きく変化し従来のワークステーションを中心とした固定的な運用形態に加えて、ノートやサブノートと呼ばれる小型で携帯可能な計算機が登場している。一方、従来のオペレーティングシステムでは、これら携帯型の計算機を十分に生かせるだけの機能を用意していない。

携帯型の計算機で従来までのオペレーティングシステムを運用することを考えた場合、あまり考える必要のなかった、様々な制約が存在している。例えば、移動するノードを考慮したネットワーク機能や分散ファイルシステム、IC カードといった付加的なデバイスのサポート、バッテリーによる運用を考慮した細かなパワーマネジメント機能などが考えられる。ここでは、現在のオペレーティングシステムにおける実装の現状とその将来性について述べる。

最後に新しい計算機コンセプトとしてのモジュール結合携帯型計算機とそれに対応したオペレーティングシステムの概要について述べる。

第 2 章

4.4BSD とその派生 OS について

今までは Internet の研究者が試験実装や実際のリリースのために用いる OS としては、歴史的経緯やコードの質の問題から、UCB(University Carifornia Berkley) の CSRG(Computer Systems Research Group) がリリースする BSD-UNIX:Berkeley Software Distribution UNIX が多く使われてきた。それらは、歴史とともに 4.2 BSD、4.3BSD、4.3BSD-tahoe、4.3BSD-reno、4.4BSD と推移してきており、今なお、BSD-UNIX は Internet 研究のベース OS として中心的な存在といえる。

しかしながら、これらの OS を開発・実験に用いるためには、USL(過去には AT&T、現在ならば X/Open) との間に V32 以後の UNIX ソースライセンス取得契約を締結し、高価なライセンス料を支払い、守秘義務を結んだうえでなければ利用することが不可能であった。コストの問題はともかく、守秘義務の問題は、特に学術研究において、自由な論文発表、ノウハウの移転を決定的に妨げてきたため、多くのネットワーク研究者は、UCB が 4.3BSD-tahoe からそのネットワークコード部分だけを取り出して Freeware 化したソースコード集 BNR1(Berkley Network Release/1)、あるいは、4.3BSD-reno から実に 85%を切り出してきた BNR2 によってすくなくともネットワーク部分のコードに対する自由な改造実験・論文発表を行なってきた。ところが、近年、BNR2 をベースに独自の拡張によって完全に USL ライセンスフリーであることからソースコード配布自由であり、しかも、安価な AT 互換機で動作する BSD-UNIX[137] が登場し、研究者に非常に大きな自由が与えられたことは特筆に値する。

BNR2 は USL と UCB の間の訴訟合戦のため、配布中止となったが、4.4BSD の 90%を取り出した 4.4BSD-Lite(BNR3) をベースに、現在、以下のような USL ライセンスフリーの BSD 系統の OS[138] が入手可能となっている。

- NetBSD
- FreeBSD
- BSD/OS

特に現在 WIDE Project では、研究用のベースとして、主に BSD/OS を多用しており、NetBSD、FreeBSD も必要に応じて利用している。そこで、本章ではこれらの OS の特徴についてまとめるものとする。

2.1　4.4BSD (encumbered)

まず最初に USL ライセンスが必要なオリジナルの 4.4BSD(encumbered) について簡単に特徴をあげておく。

4.4BSD(encumbered) の特徴

ターゲット機種 (バイナリとソース)

- HP9000/300 68000 ベースワークステーション
- DECstation 3100 および 5000 MIPS ベースワークステーション
- Sparcstation I & II SPARC ベースワークステーション (SS10 では動作しない)

以下の機種についてはソースコードが (一部) 附属

- Intel 386/486 (ISA/AT または EISA)
- Sony News MIPS ベースマシン
- Omron Luna 68000 ベースマシン

特徴としては以下のような新機能 (4.3BSD との比較) がある

- (Mach OS に由来する) 新しい仮想記憶機構
- ISO/OSI ネットワークサポート
- スタッカブルファイルシステムをサポートする仮想ファイルシステム
- Sun 互換だが再配布可能な NFS
- 新しい種類のローカルファイルシステム:LFS(Log Structured Filesystem)
- ファイルシステムを 64bit ベースに拡張
- セキュリティ対策とシステム管理を拡充
- IEEE Std 10003.1 (POSIX) に準拠
- IEEE Std 10003.2 のかなりの部分に準拠
- C ライブラリに対して多くの機能が追加
- カーネルソースがアーキテクチャ別のものと共通のものに分離
- プロセス管理の改善
- (コマンドを含む) ソースコード全体の ANSI C 対応

2.2 4.4BSD-lite ベースの OS

4.4BSD-lite ベースの OS としては、

- NetBSD
- FreeBSD

は、Freeware であり、BSD/OS は、商品である。

BSD/OS の一部の部分を除き、すべてソースコードが入手可能であり、自由に配布・利用することができる。

本節ではこれらの OS について個別に紹介を行なう。

NetBSD

UCB の Chris G. Demetriou を中心とするグループが開発を行ない、i386 のみならず、Sparc や mac アーキテクチャでも動作する OS として配布している Freeware。

現在の (オフィシャルリリース)version は 1.0 であるが、常に最新のソースコードが-current version として ftp.netbsd.org から入手可能 (国内では ftp.iij.ad.jp をはじめとする各地に mirror 有)。

4.4BSD の機能に加え、

- SUN の NIS と互換の YP
- Shared Library 機構

などが付加されている。

性格的には、ユーザーの利便よりも OS としての機能を拡充することに開発の力点が行なわれており、4.4BSD-lite ベース OS の中では常に早く新機能が搭載される傾向がある。その反面、動作ハードウェアを拡充することや、インストールツールを使いやすくする、といった傾向は無視されがちである。

なお、SPARC 用のバージョン、NetBSD/Sparc に対しては次世代 IP の IPv6 のサンプルインプリメンテーションが世界ではじめて行なわれた。

FreeBSD

freebsd.org をベースに活動するグループが配布する、やはり Freeware ベースの 4.4BSD-lite base OS である。基本的に CD-ROM で配布を行なうことを念頭に、初心者にも取り扱いが簡単なように配慮されたキット構成を取る。NetBSD で追加された新機能も後を追うようにして取り込むなど、安定性と取り回しの容易さを狙う傾向にある。

NetBSD とは異なり対象ハードウェアは i386 アーキテクチャの AT 互換機に絞られている (日本のチームによる PC98 シリーズへの移植あり)。

やはり、ftp.freebsd.org(やはり国内で mirror されている) より、最新のバージョン (-current) が常に入手可能である。現在のオフィシャルリリースの最新は 2.0.5 であるが、もうすぐ 2.1 が登場する模様である。

先端的な部分を除く NetBSD のほぼすべての機能に加え、

- gziped バイナリ実行機能
- PCMCIA サポートの標準キットへの添付

など、PC ユーザーにとって大変便利な OS となっている。

BSD/OS

米国 BSDI(Berkley Software Development Internationa) 社が開発する BSD/OS は、そもそもは、NetBSD および FreeBSD と元を同じくする 386BSD をベースに開発されてきた商品である[1]。

BSD/OS 2.0 は、やはり 4.4BSD の機能に加え Sheared Library に対応するなどしている[2]。現在のバージョンは 2.0 であり、Xinside 社の Accelerated-X を標準添付したり、制御方法が公開されないボードであってもベンダーと契約の上で、ソースコードなしのデバイスドライバを提供することでユーザーへの利便を計っている。サポートアーキテクチャは現在のところ i386 ベースの AT 互換機のみであるが、将来のリリースでは、SPARC アーキテクチャにも対応するとアナウンスされている。

各種の PC-UNIX の中で同期インターフェースに対応した唯一の PC-UNIX であり、高速な専用線に対するルーターとしても利用可能であり、実際に WIDE 内においても利用している。

BSD/OS は Freeware ではなく、商品であり、安定性、サポートなどが他の OS に比べて抜群に良いため、研究目的には最適な OS であると考えられており、特に WIDE Project ではさまざまな実験・開発をこの OS をベースに実行している。

[1] 日本ではフォア・チューン社などが取り扱い

[2] NIS に関しては対応していないが、日本国内で FreeBSD より移植された YP キットがある。

第 3 章

携帯計算機向け OS の現状と将来

携帯計算機の登場により、オペレーティングシステムは大きな変格を迫られている。これまでのオペレーティングシステムでは、

- CPU に接続されているハードウェアインタフェースは電源を入れた時点から電源を切る時点まで変化しない
- 電源は途中で落ちない
- CPU に供給されるクロックの周波数は一定

などの前提が置かれてきたが、これらは携帯計算機では必ずしも恒真ではない。多くの携帯計算機には PCMCIA スロットなどの活線挿抜可能なハードウェアインタフェースが付属している。このような携帯計算機では、CPU の電源を落すことなくハードウェアインタフェースが交換される可能性がある。電源の制御についても、バッテリ消費を最小限に抑えるために APM(Advanced Power Management) などのためのハードウェアが多くの携帯計算機に実装されている[1]。APM はキー入力が少ない場合にはクロック周波数を落す、一定時間待ち状態が続いたら液晶画面の電源を落しディスクの回転を止める、などの操作を行う。

このため、携帯計算機上で動作するオペレーティングシステムではこれまでと全く異なるシステム構築ストラテジやスケジューリングポリシが要求される。本章では、現在行われている PCAT 互換携帯計算機向けの UNIX 系オペレーティングシステムの改良、および研究活動についてまとめ報告する。UNIX ベースの実装としては主に今日からでも実際使えるものについて調査した。研究活動については今後の研究に継るようなものについて調査した。

3.1 UNIX 上での実装の現状

ここでは、自由に入手し利用することのできる、あるいはもうすぐ自由に利用できるようになる、UNIX ベースの実装についてまとめる。それぞれについてサポートするハード

[1] EnergyStar 計画 (計算機の電力消費を削減する米国政府のプロジェクト) などの進展により、デスクトップ機にも実装されている場合がある

ウェア種別とそのアーキテクチャについて述べる。
技術的に特に注意すべき点は以下のとおりである:

APM BIOS の初期化方法: ほとんどの携帯計算機には、パワーマネジメントチップセットの制御を行うための BIOS が実装されている。APM BIOS を初期化するためには、一度リアルモード (8086 互換モード) で BIOS の初期化を行なわなければならない。このためには、以下のふたつの方法がある:

- ブートレコードを書き換え、プロテクトモード (80386/486 独自モード) に移行する前に初期化を行う
- カーネル内でプロテクトモードからリアルモードへ移行し、初期化を行い、再度プロテクトモードへ復帰する

APM BIOS のインタフェース: APM BIOS をカーネル外から操作するためのプログラミングインタフェースはどのように提供されているか。

PCMCIA チップセットのアクセスインタフェース: PCMCIA チップセットをアクセスするためのプログラミングインタフェースはどのように提供されているか。通常、デバイスドライバからのアクセスのためのカーネル内関数と、カーネル外からアクセスするための IOCTL インタフェースの両方が必要である。

PCMCIA カードの初期化: PCMCIA カードには、CIS(Card Information Structure) と呼ばれる、カードの属性を記述したデータ構造がある。PCMCIA カードを利用するためには、CIS のデータに従い PCMCIA チップセットを初期化する必要がある。

初期化コードの実現方法としては、デバイスドライバ内に埋め込む、なんらかの CIS タプル解析スクリプトで実現する、といった方法が考えられる。

3.1.1 wildboar project

開発者: 北陸先端大の篠田助教授(shinoda@jaist.ac.jp) と dit の徳川氏(toku@dit.co.jp) を中心に、WIDE 研究者を中心とする協力者によって開発が行われている。

配布: 現在は大域的には配布されていない。beta-test のための配布である。将来的には配布自由となる予定である。

実現方法: APM の初期化のためにブートレコード (512byte boot) の書き換えを要する。APM BIOS をアクセスするための IOCTL インタフェースが提供されている。

PCMCIA カードの初期化コードは各デバイスドライバ内に記述せねばならない。PCMCIA カード制御のためのソケットサービス/カードサービスがカーネル内のサブルーチンとして提供されている。アプリケーションから PCMCIA カード制御を行うための IOCTL インタフェースが提供されている。

サポートされている PCMCIA カード: 図 3.1参照。

動作確認されている機種:

- Toshiba T3400、Dynabook/V486A および Dynabook/SS
- NEC Versa S/50
- Gateway 2000 HandBook 486 DX2/40 および ColorBook
- IBM TP230Cs および TP500
- EPSON Endeavor
- Fujitsu FMV-450
- Dell Latitude XP
- DEC HiNote Ultra
- Chaplet iLUFA350 486DX4/75

今後の方向: 今後は WIDE project の研究活動のためのプラットフォームとして利用されることが期待されている。例えば、フラッシュメモリディスクや電池による駆動などの特性をを活かした低消費電力のディスクレスルータの開発などが計画されている。

備考: NetBSD 用の移植が存在する。

3.1.2 LP: Laptop Package

開発者: 慶応義塾大学の細川達己氏 (hosokawa@mt.cs.keio.ac.jp) を中心に、主として bsd-nomads mailing list(bsd-nomads@ai.cs.fujitsu.co.jp) で議論と機能拡張が行われている。

配布状況: FreeBSD の標準配布にマージされている。

実現方法: APM の初期化はカーネル内でプロテクトモードからリアルモードへ一時的に復帰して行われる。APM BIOS をアクセスするための IOCTL インタフェースが提供されている。

PCMCIA カードの初期化コードは各デバイスドライバ内に記述せねばならない。PCMCIA カード制御のためのソケットサービス/カードサービスがカーネル内のサブルーチンとして提供されている。アプリケーションから PCMCIA カード制御を行うための IOCTL インタフェースが提供されている。

サポートされている PCMCIA カード: 図 3.1参照。

動作確認されている機種: 不明。

備考: NetBSD 上に Laptop Package の APM ドライバを移植したものが存在する。鵜飼文敏氏 (ukai@hplj.hpl.hp.com) 氏によるものであるが、配布は大域的には行われていないようである。

3.1.3 Linux PCMCIA support package

Linux の PCMCIA カードサポートについては、配布元の anonymous FTP サイトが移動したため発見不可能であった。このため、APM の部分についてわかっていることのみをまとめる。

開発者: APM の部分にはふたつの異なる実装がある。

(a) Stephen Rothwell 氏 (Stephen.Rothwell@pd.necisa.oz.au) によるもの

(b) Theodore Ts'o 氏 (tytso@mit.edu) によるもの

以下ではそれぞれを「APM サポート (a)」「同 (b)」と呼ぶ。

配布状況: 標準配布と同じ anonymous FTP サイトから配布されている。

実現方法: APM サポート (a) では、APM BIOS アクセスインタフェースはカーネル内に、パワーマネジメント制御は daemon プロセスとして実現されている。

APM サポート (b) では、APM 制御を行う全ての部分は daemon プロセスとして実現されている。(b) は APM bios を一切利用せず、直接パワー制御チップセットへのアクセスを行っている。このため、移植性/汎用性は非常に低いと思われる。

サポートされている PCMCIA カード: 不明。

動作確認されている機種: 不明。

3.1.4 Solaris 2.x for nomads

開発者: 米サンマイクロシステムスの Nomadic systems group で開発が行われている [139]。

配布状況: 不明。

実現方法: PCMCIA/APM サポートはカーネル内で実装されている。DDI/DKI(SVR4 系のデバイスドライバのインタフェース) にパワーマネジメントや活線挿抜関連のプロトコルが追加されている。

サポートされている PCMCIA カード: 不明。

動作確認されている機種: 不明。

備考: ネットワーク接続をシリアル回線経由の ppp と Ethernet カードの間で動的に切り替える機構が実装されている。

APM によるサスペンド (動作の一時中断) はユーザプロセスへシグナル SIGFREEZE として通知される。これを利用してユーザプロセスはネットワーク接続の切り離し、重要なデータの保存などの操作を行うことができる。

3.1.5 その他

NetBSD-current team の内部に PCMCIA ドライバを実装しているグループが存在する。NetBSD 0.9 には DE-650 と ne2000 コンパチブルの PCMCIA カードのためのデバイスドライバが付属していたようであるが、配布キットを発見できなかったため詳細は不明である。

図 3.1: サポートカード一覧

card	wildboar	LP
IBM PCMCIA ethernet card I/II	○	○
他の ne2000 互換 ethernet カード	○	×
3Com EtherLink III PCMCIA ethernet card (3C589)	○	○
3C589 互換 ethernet カード	○	×
D-Link DE650 PCMCIA ethernet card	○	○
DE650 互換 ethernet カード	○	○
Xircom Credit Card NetWave adapter	○	×
Proxim RangeLAN2/PCMCIA	○	×
MegaHertz XJ1144/2144 PCMCIA modem card	○	○
他のモデムカード一般	○	×
IBM PCMCIA Serial IR card	○	×
BUG LinkBoy D64K PCMCIA ISDN adapter	○	×
SunDisk および ATA 仕様メモリカード	○	△
New Media Corporation BusToaster PCMCIA SCSI card	○	×
I/O Magic FOCUS PCMCIA Video Capture card	○	×

3.2　研究活動

ここでは、携帯計算機の特殊事情をサポートするためのさまざまな研究活動をサーベイする。

3.2.1　Apertos オペレーティングシステムでのデバイスドライバ構築法

[140] では、Apertos オペレーティングシステム上で活線挿抜可能なハードウェアに正しく対処できるデバイスドライバの構築法について論じている。Apertos はかなり特殊なオペレーティングシステムであるので、まずその概要を述べ、その後この研究の詳細について述べる。

Apertos の概要

Apertos はソニーコンピュータサイエンス研究所で研究開発が行われている超広域分散環境のためのオブジェクト指向 OS である。「オブジェクト指向 OS」を標榜する OS としては他にいくつかあるが、それらの多くは以下のような OS である:

- オブジェクト指向に基づいて設計実装されるアプリケーションを支援するための OS カーネル。OS カーネル自身は単層カーネル構成 (Chorus[141] や Mach[142, 143] など)。
- C++によって記述された OS カーネル。実際には C++オブジェクトは静的に結合され、単層カーネルを構成する (Choices[144] など)。

Apertos ではそれらのアプローチと異なり、OS を実現する全ての部分を並行オブジェクトとして実現しようとしている。すなわち、単層の OS カーネルは存在せず、これまでの OS で単層カーネル内で実現されていたスケジューラ、仮想記憶マネジャなどは全て独立した並行オブジェクトとして実現される。

しかしながら、並行オブジェクトとして実現されたシステムプログラムやユーザアプリケーションを、実際に実行する際には、それぞれの役割によって異なる実行メカニズム、異なる通信セマンティクス、異なる保護機構が要求される。例えば、複数ホスト間で互いに通信を行うユーザアプリケーションは位置透過な通信機構を必要とするが、位置透過な通信機構を実現するためのシステムプログラムは、位置に依存した通信機構を用いることによってしか実現できない。また、プログラムの実行速度とメモリ保護のどちらを優先するべきかなどはプログラムの種類、プログラムの記述言語や言語処理系などによって異なる。Apertos では自己反映計算の枠組を用いて、並行オブジェクトによる実現とさまざまな実現形態を両立させている。

Apertos では、各オブジェクトはメタ空間と呼ばれる実行環境上で実行される。メタ空間は各オブジェクトに対し、メタコールと呼ばれるシステムコールのような仮想命令セットを提供し、各オブジェクトはそのメタコールを介して他のオブジェクトとの通信やメモリ

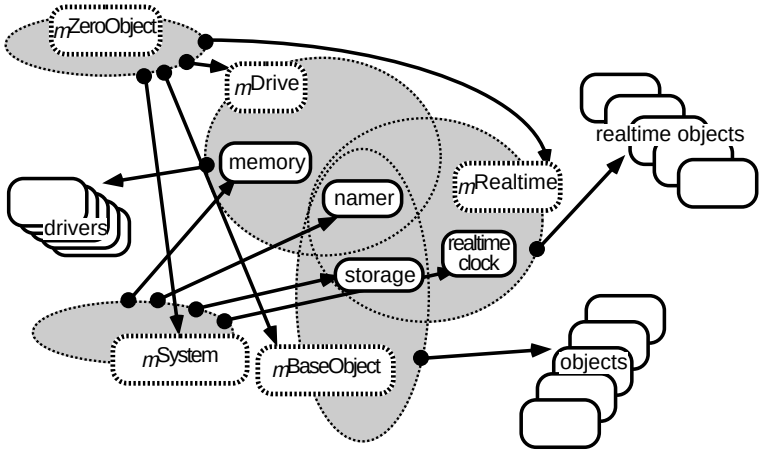

図 3.2: Apertos におけるメタ階層

割り当てを行う。メタコールの実現はメタ空間によって規定されるので、各オブジェクトの実現に適したメタ空間を準備することで、オブジェクトの質にあわせた実行を行うことができる。メタ空間を実現するのはオブジェクト群 (メタオブジェクト) であり、オブジェクトとメタオブジェクトの関係は相対的である。

図 3.2において[2]、グレーの円がメタ空間を、楕円の点線がオブジェクトを表している。メタ空間 m^{Drive} 上にはドライバオブジェクトが複数個存在しており、m^{Drive}を構成するオブジェクト群として memory メタオブジェクトと namer メタオブジェクトがある。memory メタオブジェクトはメタ空間 m^{System} 上で動作している。ドライバオブジェクト群はメタ空間 m^{Drive}に対しメタコールを起こすことでオブジェクト間メッセージ送信やメモリ割り当てなどの操作を行う。メタコールは memory メタオブジェクトによって実際に実行される。

Apertos におけるデバイスドライバの実装

先にも述べた通り、[140] では、Apertos オペレーティングシステム上で活線挿抜に対応してインストール/除去できるデバイスドライバを作るための条件について論じている。そのためには、少なくとも

- 各デバイスドライバの間に共有されるデータがないこと
- 各デバイスドライバが互いに直接影響されないこと、特に排他制御のために他のデバイスドライバの実行を強制的に止めたりしないこと

[2]図は現在の実装と若干異なる

が必要であるとし、各デバイスドライバを独立した並行オブジェクトとして実現している。各オブジェクトはメッセージ通信により情報交換を行い、一切共有データを持たない。また、排他制御はオブジェクト単位で自動的にメタ空間によって行われ、コード中に手で記述する必要は一切ない。割り込み処理の要求はオブジェクトへオブジェクト間のメッセージと同様に配送され、プログラマは両者を区別する必要がない。実行安全性はデバイスドライバのためのメタ空間によって保証される。このように実現を行うことで、各デバイスドライバオブジェクトを実行中に動的に交換することが可能となる。

類似の研究・実装としては UNIX LKM などがあるが、これらはデバイスドライバ間の排他制御や割り込み待ちの検出を行わずにデバイスドライバをメモリ中から除去するようになっているため、タイミングによってはシステム全体の停止を索き起こす場合があり得る。Apertos で行われている方法では、このような障害は発生しない。

3.2.2　携帯計算機のためのスケジューリング

[145] では、携帯計算機のためのプロセススケジューリングポリシについて論じている。これまでの計算機でのスケジューリングポリシは種としてプロセス間の公平性や実時間制約などを重視していたが、携帯計算機の場合には電池利用量などが重要なスケジューリングの基準となる。このため、この論文では MIPJ(Million Instruction Per Joule) という評価基準を導入し、複数のスケジューリングポリシ間の比較を行っている。

3.3　その他

近年携帯計算機に関する研究は多数行われており、他にもファイルシステムやネットワーク層プロトコルなどに関する研究が多数存在する。USENIX OSDI'94 などに多数論文があるので参照されたい。

第 4 章

通信機能を重視した携帯型計算機用オペレーティングシステム

4.1 はじめに

インターネットに代表されるコンピュータネットワーク環境はここ数年飛躍的な発展を遂げている。一方、高性能な携帯型計算機の登場は、計算機環境に大きな変化をもたらしている。このような環境では、従来までの固定的な運用形態に留まらず、ネットワークを含む様々なデバイスがダイナミックに生成/接続/消滅していく中でシステムが運用されることになる。

我々は携帯型計算機とそれを取り巻くネットワーク環境・オペレーティングシステムについて概観し、ネットワークと携帯型計算機を結びつけた新しい計算機アーキテクチャとして、モジュール結合携帯型計算機を提案している。そしてモジュール結合携帯型計算機用のオペレーティングシステムである T-OS を提案する [147]。ここでは T-OS の概要を述べ、オペレーティングシステムを構成する各要素について述べる。

4.2 モジュール結合携帯型計算機

現在の携帯型計算機システムの多くは、重量や稼働時間などに制約が存在している。これは計算機を使用するのに必ずしも必須とはいえない、キーボードやディスプレイといった周辺機器が付属しているためである。そこでシステム全体を、いくつかのモジュールに分割し、ユーザの必要に応じて組み合わせて利用する計算機を考える (図 4.1)。これをモジュール指向携帯型計算機と呼ぶ。

例えば普段は必要最低限の機能を用意したメインモジュールだけを携帯する。ユーザは必要に応じてモジュールを追加し、機能拡張していく。例えば、ディスクモジュール/ディスレイモジュール/ネットワークモジュールといったものを挙げることができる。

それぞれのモジュールはシステムバスによる接続だけでなく、ネットワークなどを使って相互接続されている場合もある (図 4.2)。

周辺機器を追加変更できるだけなら、従来でも似たようなシステムは存在している。例

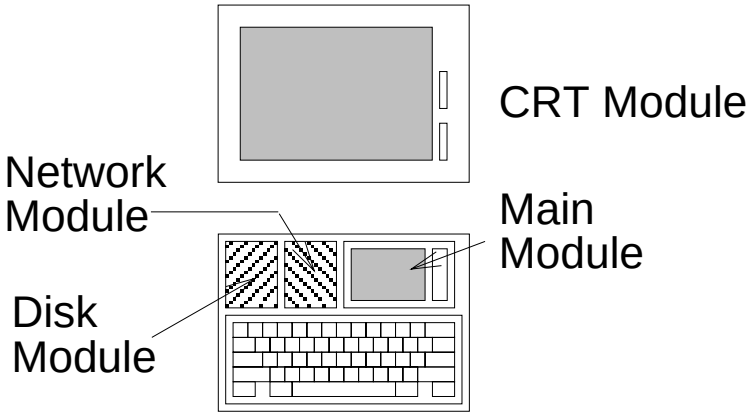

図 4.1: モジュール結合携帯型計算機外観

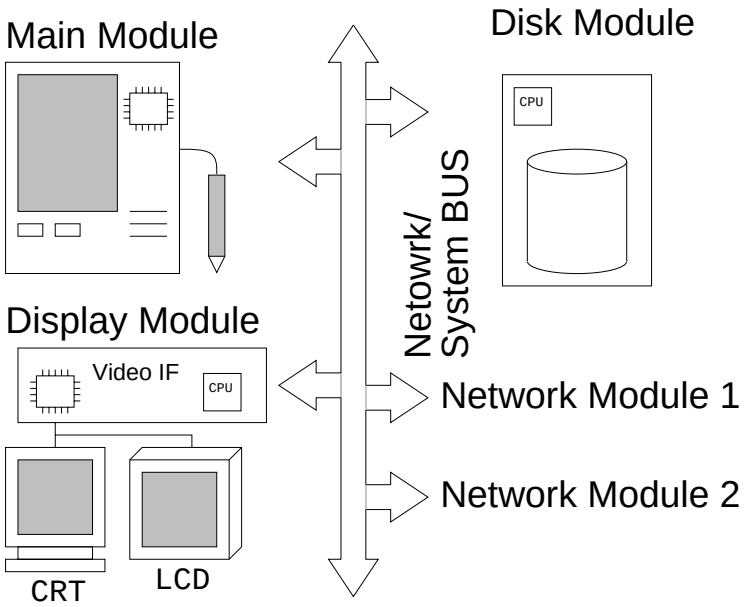

図 4.2: システム構成

えば一部のノートパソコンに用意されている “ドッキングステーション” と呼ばれているものがそうである。

ここで重要なのは、分割された個々のモジュールは単体でも動作し、運用可能である点である。例えばディスクモジュールはファイルサーバの形でメインモジュールからネットワーク等を介して利用可能にする。これにより全体の機能を削減することなしに自分の要求に応じて動的にシステムを変更しながら運用することが可能になる。

さらに、同等の機能を提供する複数のモジュールが同時に接続されて運用することも考える必要がある。例えば複数のディスレイモジュールを同時に接続し、それぞれ違う内容の画面を表示させて使用するといった運用にも適用できる。

4.3 T-OS

モジュール結合携帯型計算機を有効に利用するためにはオペレーティングシステムのサポートと、これに対応したアプリケーションの存在が必要不可欠である。

4.3.1 オペレーティングシステムに要求される機能

モジュール結合携帯型計算機の特徴は、複数のモジュールがシステムバスやネットワークを介して結合/切断され、システムの構成が動的に変化することである。そのためオペレーティングシステムはモジュールの動的な再構築機構を用意する必要がある。またモジュール間はシステムバスやネットワークが介在するため、情報のやり取りは基本的にメッセージを用いた通信となる。この時、情報伝達手段も状況に応じて変化するので、ネットワーク構成の動的な再構築機能を用意する必要がある。

4.3.2 構成

モジュール結合携帯型計算機が動作するためには、T-OS がそれぞれのモジュールで動作している必要がある (図 4.3)。これは構造的には疎結合マルチプロセッサに近いが、モジュールの構成がユーザの使用する環境やネットワークの変化によって様々な形に動的に変更される点で異なる。

メインモジュールのような単体で動作可能なモジュールの場合、次のような機能を用意する必要があると考える (図 4.4)。

- カーネル
- スケジューラ
- ネームサーバ
- 基本デバイスドライバ

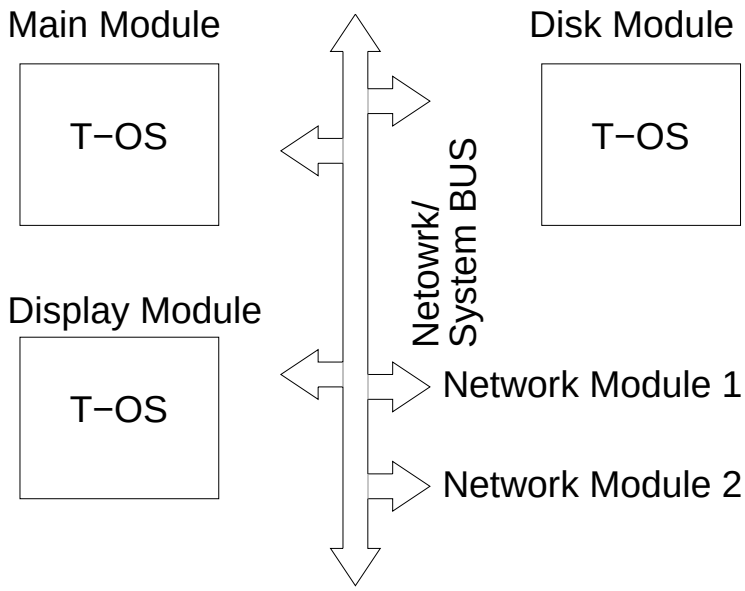

図 4.3: T-OS:システム

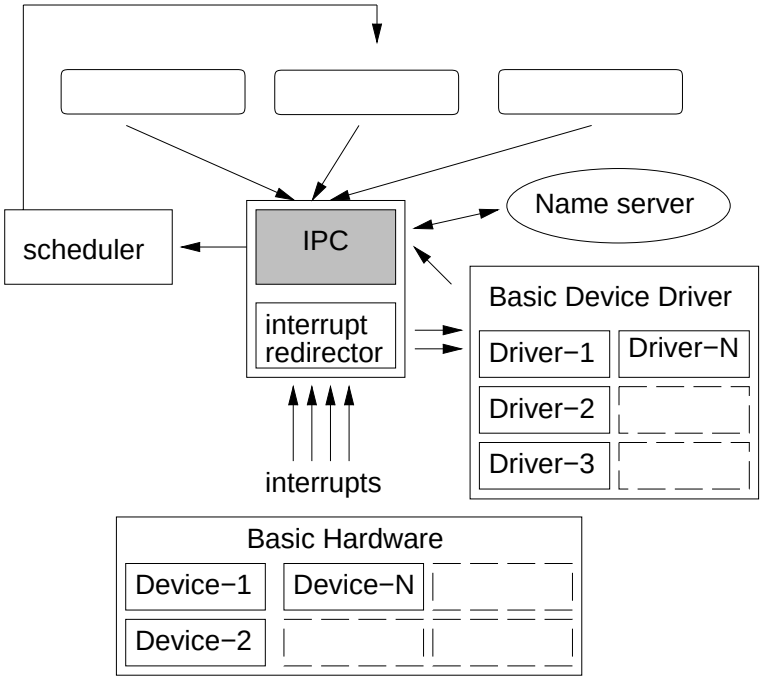

図 4.4: T-OS:構造

カーネルは、ハードウェアなどからの割り込みを対応するドライバに割り振る interrupt redirector とモジュール内の各種プロセス間のメッセージ伝達を制御する IPC から構成される。

基本デバイスドライバは、オペレーティングシステムに必要な各種デバイスドライバのインターフェースを提供する。

ファイルシステムは必須ではないが、デバイスドライバなどをロードするためにはこれを扱えるサーバが必要になる。

ネームサーバはそのモジュールが用意するサービスの情報やポインタを管理し、プロセスから情報の登録/変更/削除などを可能にする。

4.4 プロセス間通信

T-OS ではメッセージ伝達によるプロセス間通信機構を用意するが、これはモジュール内外のあらゆる通信に用いられ、システムの性能にもっとも影響を与える部分となる。

モジュール結合携帯型計算機環境では、モジュールやネットワークの動的な再構築を伴い、メッセージの送受信先が必ずしも特定出来ない場合がある。そこで Mach[148][146] のような間接型のメッセージ伝達手段が必要となるが、Mach では間接指定された送信先をカーネル内部で決定するため、メッセージの転送先をカーネルは常に意識している必要がある。そのため通信相手のシステム構成が動的に変化するようなシステムに適用するには、複雑なメッセージの型のチェックなども考慮するとオーバーヘッドがかなり大きくなることが予想される。

上記の問題を解決するために、T-OS のプロセス間通信には簡単な型付の固定長メッセージを採用する予定である。これにより型の種類を減らし、固定長メッセージを用いることで構造チェックなどによるカーネルのオーバヘッドを抑えた高速な通信を実現する。現在は型の種類や最適なメッセージサイズなどについて検討中である。

また動的に変化するモジュールへの通信機能の為に、マルチキャスト的な通信機能を用意することを考える。送信側は特定のモジュール群を代表するマルチキャストグループに対してメッセージを送信する。カーネルは受信側の状態を意識する必要はなく、メッセージの処理だけを行うため、相手先を決定するための複雑な処理を軽減することが可能になる。またマルチキャストを使えば複数のモジュールに同時にメッセージを伝えることも可能になる。現在はマルチキャストメッセージ通信の具体的な有効性について検討中である。

4.5 ネットワーク機能

T-OS ではネットワークドライバやプロトコルスタックの動的な変更を含めたネットワーク構築機能を実現していく (図 4.5)。

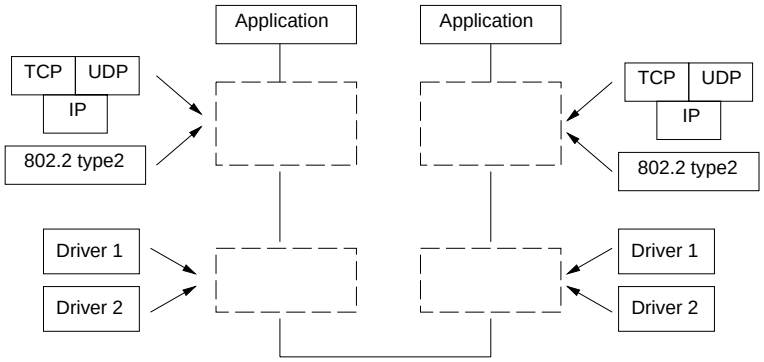

図 4.5: ドライバ/プロトコルの動的変更機構

携帯型計算機で提供されるデバイスの多くは、標準的なドライバを用意している。例えば多くのイーサネットカードは、特定のオペレーティングシステム向けではあるが、共通インタフェースを提供するドライバが用意されている。T-OS では本来別のオペレーティングシステム用のドライバでも選択的に利用可能な機構を用意する予定である。これによってドライバ構築のコストを下げることが可能になる。

次にデータリンクレベルに応じて途中のプロトコルスタックを変更したり、省略するような機能を用意する。具体的にはデータリンクレイヤで信頼性のある通信が確保されている場合はトランスポートレイヤを省略することでプロトコル処理のオーバーヘッドを減らすといったことを可能にする。これを実現する為、プロトコルの階層間のネゴシエーション規則などについて検討中である。

4.6　名前規則/資源検索

T-OS からある資源を参照したり、通信をするといったことを考える場合、相手を指定するための規則を規定する必要がある。UNIX ではファイルシステムを用いた指定が一般的だが、T-OS では新たに、別の名前規則を導入することで解決する。これはネームサーバに登録したリソースを基本としたものとする。

ネットワーク上のリソースを指定する手段として URL [113] が最近よく用いられている。URL をそのままネーミングポリシーとしてシステムに導入するには無理があるが、これを参考にしたものを導入していく。特にサービスによっては、それを用意している近くのモジュールが利用できれば良い場合もある (例 かな漢字変換サーバなど)。このように相手のモジュールを特定しない曖昧な指定手段を用意することでサービス指向のリソースアクセス機構を実現する。