分散型個人用計算環境のためのUNIXの自然な拡張の試み†

宇津宮 孝一††* 清水 泰行††**

単一計算機のオペレーティング・システムとして開発されたUNIXは，近年，個人用計算機やワークステーションに共通のプログラミング環境を提供するものとして，その重要性をますます高めている．我々は，UNIXを異種性の高い分散型個人用計算環境に自然に拡張することを試みた．この拡張によって，ローカルエリア・ネットワークで結合された他の異機種UNIX計算機の資源を，一様な利用者インタフェースで利用できる．しかも，従来のシェルを基盤にして拡張を行ったので，カーネル等の変更を伴わず，拡張部分の異種UNIXへの移植性はもちろん，既存のソフトウェアの再利用性を著しく高めることができる．本論文では，重要な基本機能の実現，その分散型計算環境への適用，効率，および移植性や再利用性などプロトタイピングでの経験について述べる．さらに，自律的に動くUNIXシステムがネットワークで結合された分散型計算環境における本接近法の有効性とその問題点についても議論する．

1. はじめに

高性能個人用計算機やワークステーションの出現は，1台ですべてのことができるとの錯覚を与えがちであるが，計算機間の情報の交換やデータベースなど有用な資源の利用は，従来以上に重要となっている．しかし，自律性，異種性（heterogeneity）の高い計算機同士を，個人の計算環境の自然な拡張として互いに接続することは，それほど容易ではない．

過去に我々は，異機種計算機の対等な結合を行う「対称型ソフトウェア」の概念を提案した[1]．当時の実験環境（オペレーティング・システム（以下OSと略）や伝送媒体など）は，非常に貧弱で，既存ソフトウェアの移行も困難を極めたため，概念の有効性の確認はできたものの，ネットワーク透過性の高い利用者インタフェースの実現までには至らなかった．これは，その後のUNIX***やローカルエリア・ネットワーク（LAN）の発展・普及を待たねばならなかった．

中央計算機としての種々の機能を果たす汎用計算機と，同種OSで自律的に動く異機種の高性能個人用計算機が，LANで疎結合された研究者向け分散型計算環境の構築を，上述の概念を基盤にして行っている[2]．この一環として，新たな分散型OSの構築を念頭に置きながら，UNIXに基づくプロトタイピング（UNIQ-NETと呼ぶ）を試みた[3]．これは，既存のソフトウェアの再利用性[4]を高め，さらに一様な利用者インタフェースを提供できるように，UNIXを分散型計算環境へ自然に拡張したものである．しかも実現はシェル・レベルで行ったので，カーネル等の変更を全く伴わず，異機種UNIX計算機に容易に移植することができる．

本論文では，分散型計算環境における基本的な機能の実現，その適用，および有用性と問題点などについて，プロトタイピングでの我々の経験を述べる．

2. 設計概念と実現手法

2.1 設計概念

UNIQNETにおける分散型システムの形態は，図1に示すように，Wandらの分類による「ネットワークの隠蔽による自律型システムのネットワーク」の範疇に属する[5]．各個人用計算機には，必ずしも同一でない（版や出所の相違のため）UNIXが走っている．別のOSが搭載されている汎用計算機は，最小限必要なUNIXインタフェースをもつ．

本質的には単一計算機上でのみ動くUNIXを[6]，分散型計算環境に適用するためには，再設計あるいは拡張が必要である．現在までに，(a)完全に再設計したLOCUS[7]，(b)カーネルのすぐ上に新たな階層を挿入したNewcastle Connection[8]やS & Tnet[9]，(c)ネットワーク・コマンドを追加したUUCP[10]などの試みがなされてきた．完全なネットワーク透過性を達成するには(a)の接近法しかない．一方(b)は，ネットワーク透過性，効率，および利用者プログラム内からの他系（リモート・システム）アクセスまでを考慮した方法で，同種UNIXの環境では効果的である．

† Experiences with Extending UNIX Consistently to a Distributed Personal Computing Environment by KOUICHI UTSUMIYA and YASUYUKI SHIMIZU (Department of Information Sciences, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University).

†† 九州大学大学院総合理工学研究科情報システム学専攻
* 現在 大分大学工学部組織工学科
** 現在 富士ゼロックス(株)ソフトウェア技術センター

*** UNIXオペレーティング・システムは，AT&Tのベル研究所が開発したものである．

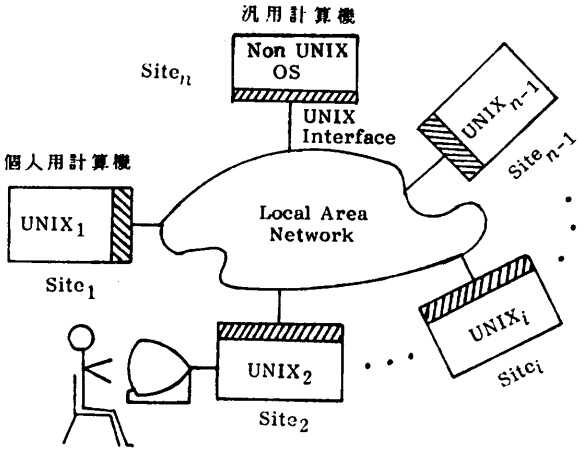

図 1 分散型個人用計算環境
Fig. 1 A distributed personal computing environment.

ただし，実現上はカーネルの一部とシェルの変更を伴う．(c)の方法は，コマンドの新規追加のみで異なるUNIX の環境に対応できるので簡単である．しかしこの方法は，ネットワーク透過性，利用者インタフェースの一様性，および他系との協同処理機能の点で問題がある．

我々の接近法は(b)と(c)の間に位置する．これは，下記の設計概念に基づいてシェル・レベルでの実現を図ることにより，各異種型個人用計算機のみに依存した機能を有する各 UNIX を，カーネル等の変更を全く行わずに，蓄積ソフトウェアの互換性を保ちながら分散型計算環境へ適応させることができるので柔軟性が高い．現時点では，第5章で述べるように使用上の制限や効率上の問題点は残るものの，本接近法は，一様な利用者インタフェース（既存コマンドと同様の使用法）とある程度のネットワーク透過性を達成できる実現が容易な，実際的な手法である．

（1） UNIX の概念の自然な拡張

既存の UNIX を，同一の概念で分散型計算環境でも取り扱えるようにする．

（2） ネットワーク透過性の高い利用者インタフェース

各計算機は別の管理下で運用されることを利用者に意識させるが，利用者が，自系（ローカル・システム）とほぼ同様に他系を使用できる利用者インタフェースの一様性は保証する．

（3） 異種性への対応

異種性の高さが従来のシステムとの大きな相違点である．したがって，ネットワークと既存 UNIX の間

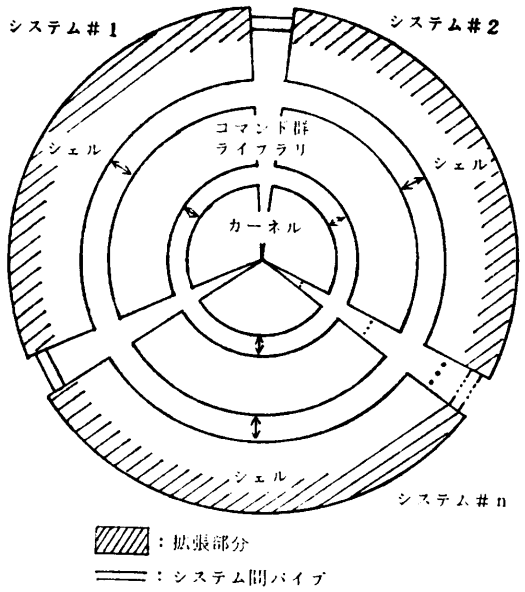

図 2 UNIX 拡張部分の位置付け
Fig. 2 An extension to UNIX.

に新たなソフトウェア階層を付加することによって，異種性，ソフトウェアの再利用性，利用者インタフェースの一様性などに対応する．また，資源使用の単位として，抽象性の高い対象であるコマンドを用いる．

（4） 基本的機能

分散型計算環境では，計算機間の接続，異なる計算機上のプロセス間の柔軟な通信，異種端末間における画面単位の転送などの機能が最も基本的である．

2.2 実現手法

（1） シェル・レベルでの実現

実現は，従来のシェルを基盤にして行い，新たにネットワーク・シェル (Nshell) を開発した（図2）．Nshell を中心とするパイプ結合の利用者プロセス群を用いて，分散化に関連するすべての機能を実現した．また，UNIX のシェル・プログラミング機能を援用することにより，Nshell のレベルで一様な利用者インタフェースの提供も行う．

（2） コマンドを基盤とする遠隔対象指向方式

UNIX のプログラミング・スタイルは，コマンドの組合せを基盤とする．したがって，他系資源を使用する場合の処理は，対象（資源：UNIX ではファイルとして抽象化される）が存在する系に，適用するコマンドとパラメタをメッセージとして送信し，その結果を受信する遠隔対象指向方式を基本とする．

（3） 動的マウント方式を用いたシステム間接続

他の幾つかのシステムでも実現されている UNIX

のマウントの概念をネットワークに拡張する.

(4) パイプを基盤とするプロセス間通信

UNIX のパイプをネットワークに拡張し，プロセス間の一方向・双方向通信や有用な分散処理機能を実現する.

(5) 画面の仮想化

仮想画面と呼ばれる分散型計算環境全体に共通の論理的画面を定義し，これに基づいて画面転送を行う.

(6) ソフトウェアの構成法

Nshell を中心に，関連プロセス群を階層構成し，それらの間はパイプを経由しての通信によって処理を進める．この構成の下で，基本的機能やその効果的応用を試作する．残りの機能および効率の問題は，当面，既存 UNIX で解決できる範囲内にとどめる.

3. 実　　現

3.1 試作システム (UNIQNET)

(1) ハードウェア構成

自律的に動く UNIX システム 3 台(UX-300, TOWER 1632 および OA-8140) と共用資源としての汎用計算機 (M-240 H) が，バス型LAN (当面 BRANCH 4800 シリアル伝送媒体 (9,600 bps) を試用) で接続されている (図 3)．これは分散型個人用計算環境としての最小構成条件を満足する.

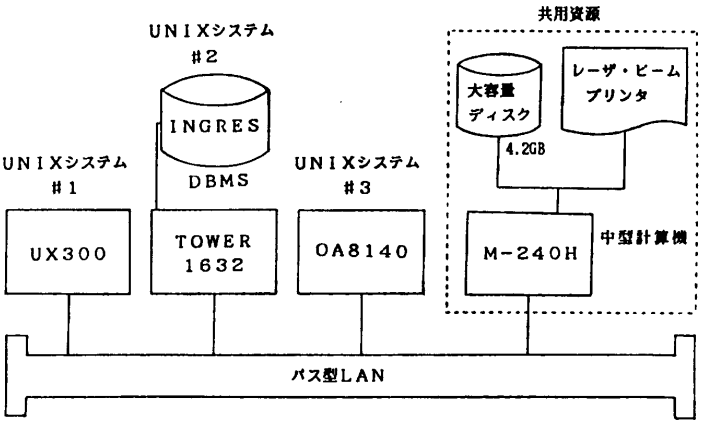

図 3 ハードウェア構成
Fig. 3 Hardware configuration of prototype system.

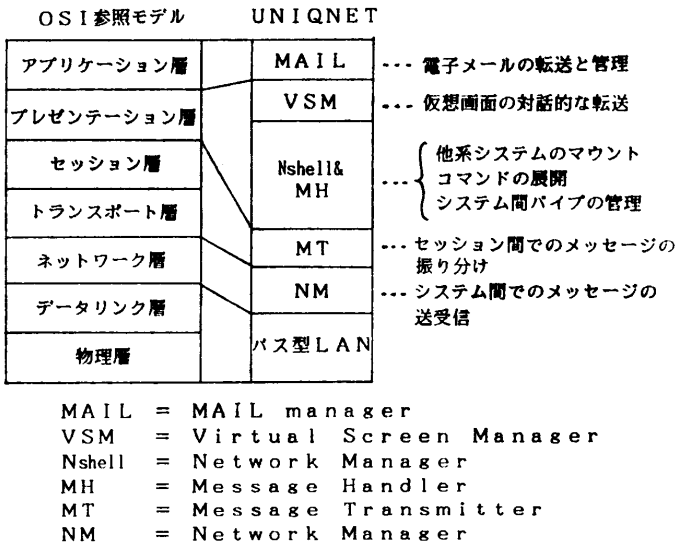

図 4 ソフトウェアの階層構成
Fig. 4 Layered structure of software system.

(2) ソフトウェア構成

各 UNIX に付加するプロセス群の構成を図 4 に示す．各プロセスの機能の概略は次のとおりである.

①NM (Network Manager) 他系の状態とシステム間でのメッセージ (データ) 転送の管理.

②MT (Message Transmitter) セッション間でのメッセージ (データ) 転送の管理.

③Nshell (Network shell) UNIX のコマンド・インタプリタである shell を基盤にして，分散型計算環境のための基本機能を付加したもの.

④MH (Message Handler) 他系からのメッセージのインタプリタで，Nshell と同じ機能をもつ.

⑤VSM (Virtual Screen Manager) プロセス間で仮想画面を転送するプロトコルを提供する.

⑥MAIL (MAIL manager) 電子メールを仮想画面として転送し，その管理を DBMS で統合的に行う.

図4の NM から VSM までの階層が，図2のシェル拡張部分に対応する．また Nshell (MH) の中から標準シェルの機能を利用する．層間のインタフェースについては，Nshell から NM までは，隣接層間のパイプにメッセージをパケット化して流し込むことによって実現する．NM と LAN 媒体との間ではパケット形式の変換が行われる．一方，MAIL と VSM 間は仮想画面化したメールの授受により，VSM と Nshell 間は双方向通信機能を用いたデータ授受により実現す

る．

（3） 基本機能

分散型計算環境のための基本機能，すなわち，

①システム間の論理的結合

②異なるシステム上のプロセス間の通信

③仮想画面の転送

は，後述するように，Nshell や MH および各機能に対応した上述の関連プロセス群によって実現される．

3.2 システム間の接続

図5に示すように，他系（例えば，UNIX System 2）と自系（例えば，UNIX System 1）を接続したい場合には，利用者は次のようにコマンドを入力する．

/etc/mount /dev/unix-system 2 /usr/u 11/mnt 1

Nshell は，これを解釈して以下のことを行う．まず，指定された他系をその利用者が使用できるかどうか，後述する利用者 id のクラスを調べる．次に，マウント・ディレクトリ（この場合は mnt 1）のアクセス権を検査する．最後に，既に他系がこの利用者によってマウント済みでなければ，自系 NM は，LAN が提供するプロトコルを用いて他系の存在を確認した後，他系 NM からの「マウント可」応答を待つ．これによりマウント可能であれば，各利用者ごとにそのシステム名とパス名をマウント表に登録する．そして見掛け上，他系ファイル・システムを自系のディレクトリ mnt1 の下に動的にマウントすることにより，他系を論理的に結合する．その後，利用者が他系のファイル f 21 をアクセスするには，

cd mnt 1/usr/u 21

により，カレント・ディレクトリ u 21 に移動し，必要なコマンドを投入すればよい．利用者がコマンドのオペランドでファイル名を指定すると，Nshell は上述した情報から実際のファイル位置を表す完全なパスを決定し，それをファイルが存在する系にコマンド名とともに送信し，実行を依頼することになる．利用者 id については，当面，自系のみ使用可能な id，ある系でのみ使用できる id，および全域的に使用可能な id の三つのクラスを用意する．

3.3 プロセス間通信

ソフトウェアの再利用性を高める重要な UNIX の機能はパイプである．パイプの概念をシステム間に拡張し，一方向・双方向システム間パイプを実現した．

一方向システム間パイプは，異なるシステム上の二つのプロセス間で，プロセスの標準出力を他のプロセスの標準入力と結合し，一方向通信を行うものである．一方，双方向システム間パイプは，異なるシステム上の二つのプロセスの間で標準出力と標準入力を互いに結合し，双方向通信を行うものである．

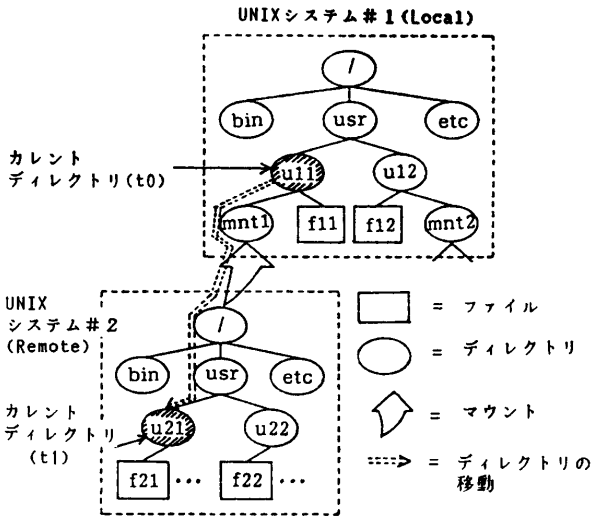

図5 システム間の接続（動的マウント）
Fig. 5 Intersystem connection (dynamic mounting).

ただし敷設可能なパイプ数は，システム間には制限はないが，同一の一対のプロセス間では一本に限る．

3.3.1 一方向システム間パイプ

システム間パイプは，**図6**（破線で囲んだ部分は除く）に示すプロセス群によって実現される．すなわち，利用者が起動したプロセス（以下，利用者プロセスと略）への入力を制御する三つのプロセス（DD, SDD および PC），利用者プロセスの標準出力を他系上の次の利用者プロセスへの入力とするためにデータ転送を制御するプロセス（SOFM），さらに，エラー・メッセージを自系のディスプレイ上に表示させるためのプロセス（SEFM）などによって構成される．

今，利用者が次のコマンド行を入力したと考える．

COMMAND 2 | COMMAND 1 | command 0

ここで，"|" はシステム間パイプを表す．また，command 2, command 1, command 0 は，それぞれ他系2，他系1，自系で起動されるコマンド（プロセス）とする（大文字表記は，このコマンド文字列のオペランドに記述された対象（通常，ファイル）がネットワークにわたるパス名を含むか，あるいはシステム名そのものであることを表す）．

以下この例を用いて，システム間パイプの敷設と実行の二つのフェーズについて説明する（図6）．

（1） システム間パイプの敷設

Nshell（他系では MH）は，各利用者プロセスの標

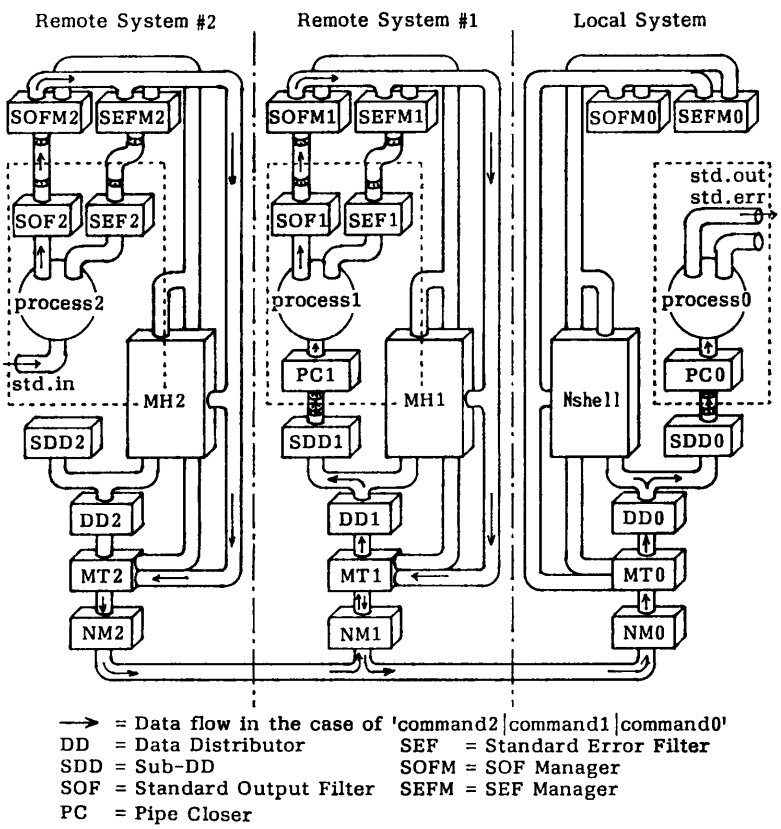

図 6　一方向システム間パイプ
Fig. 6　One-way intersystem pipe.

準出力（他系ならば標準エラー出力も）の行き先情報を管理するプロセス（SOF（他系ならば SEF も））を生成し，SDD，PC，SOFM（，SEFM）との接続のためのパイプラインを決定する．ここで，SOF 2 および SOF 1 が保持するデータ転送経路情報の内容は，それぞれ SOFM 2→MT 2→NM 2→NM 1→MT 1→DD 1→SDD 1→PC 1→〔process 1〕と SOFM 1→MT 1→NM 1→NM 0→MT 0→DD 0→SDD 0→PC 0→〔process 0〕となっている．

（2）実　　行

以上の準備がすべての系で完了した後に，Nshell は利用者プロセスの起動を行う．この後のデータの流れは，〔process 2〕→ SOF 2 ⇒〔process 1〕→ SOF 1 ⇒〔process 0〕（⇒の部分は，それぞれ前述した経路である）のようになる．

3.3.2 双方向システム間パイプ

双方向システム間パイプを利用する一対のプロセスは，下記の二つの方法によって起動される．

（1）同一利用者による起動

同一利用者が双方向通信を行う二つのプロセスを起動する場合は，一方向システム間パイプの拡張と考えることができる．この場合，起動に時間的なずれが生じないので，利用者は次のようにコマンドの起動を要求する（^^ は双方向システム間パイプを表す）．

command 0 ^^ COMMAND 1

その時，Nshell は，各コマンドを自系および他系で起動し，入力と出力のために 2 本の一方向システム間パイプをその間に敷設すればよい．

（2）異なる利用者による起動

異なる利用者が起動する場合には，時間的なずれが生じる．そこで，双方向通信をサーバ・プロセス(SP)とクライアント・プロセス(CP)の通信であると考える．利用者 1 は，まず，Nshell 1 に対して次のようにして SP の起動を要求する．

SP ^ ^

しかし，Nshell 1 は，すぐには SP の起動を行わず，CP の起動要求が出されるのを待つ．次に，利用者 2 が，Nshell 2 に対して CP の起動要求を発する．

CP ^ ^ {SP}

Nshell 2 が，Nshell 1 に CP の起動要求の発生を通知すると，Nshell 1 から Nshell 2 に許可が出され，それぞれの Nshell によりプロセスの起動が行われる．このように，サーバ・プロセスを起動する側の Nshell でランデブが起こる（図 7）．

3.4 仮想画面転送機能

仮想画面転送サブシステム VSM は，基本的には，UNIQNET が提供する双方向プロセス間通信機能を

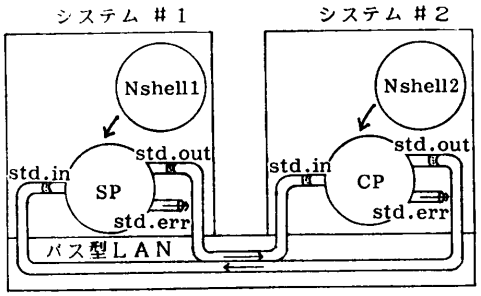

図 7　双方向システム間パイプ
Fig. 7　Two-way intersystem pipe.

利用して，二つのプロセスがもっている仮想画面の双方向転送を行うものである[11]．仮想画面は重なりのない任意の大きさの1枚の矩形で，文字，グラフィック，属性の三つの領域から構成される．文字は2バイトの仮想コード（漢字は JIS 漢字コード+16 進 '8080'，半角英数字は JIS8 コード+16 進 '0000'）で，グラフィックはビットパターンで，属性はデータ種別や画面表示形式などを属性コードとして表現したものである．UNIX 4.2 BSD の termcap 同様の装置属性表を用いて，仮想画面の任意の区画を機種に応じてディスプレイ画面上の指定場所に写像し，最終的な表示画面を得る．この目的のために VSM は，仮想画面を通常のファイルとして管理し，利用者プロセスとの入出力，仮想画面との入出力，実画面への写像，仮想画面の送受信，および実コード・仮想コード間の変換などの処理を行う．これらの機能を用いて，機種に依存しない画面指向の利用者インタフェースを容易に作成できる．

4. 分散型計算環境への適用

ネットワーク・シェル(Nshell)が一様な利用者インタフェースを提供する機構およびこのインタフェースの下で，前述したシステム間パイプなどをどのように分散型計算環境に適用するかについて述べる．

4.1 利用者インタフェース

利用者の入力コマンドを分散環境で一様に機能させるために，コマンド展開の手法を用いる．このため，UNIX のコマンドをパイプライン可能な整合コマンドとそれ以外の不整合コマンドに分類する．整合コマンドは，パイプライン上での許される位置，すなわち，左端，中間，右端に応じて，それぞれ，producer（給水）型，filter（フィルタ）型および consumer（受水）型の三つの型に分類される（図8）．

UNIX は，既存のフィルタ型コマンドをコマンド・パイプライン上で組み合わせることによって操作を行うことが基本である．Nshell は，この考えに従って，利用者の入力コマンドを整合コマンドと一方向システム間パイプの組合せとして，各コマンドがネットワーク上の当該系で実行できるように展開する．表1のコマンド展開例に示されるように，フィルタ型の cat コマンドが重要な役割を演じている．cat コマンドは，標準入力から得たデータをそのまま標準出力に書き出す．また，入出力切り換えによって，その入出力を周辺装置に容易に切り換えることもできるので，特に基本的なコマンドである．したがって UNIQNET では，cat コマンドを最大限に利用することにより，下記の有用な分散処理機能を提供する．

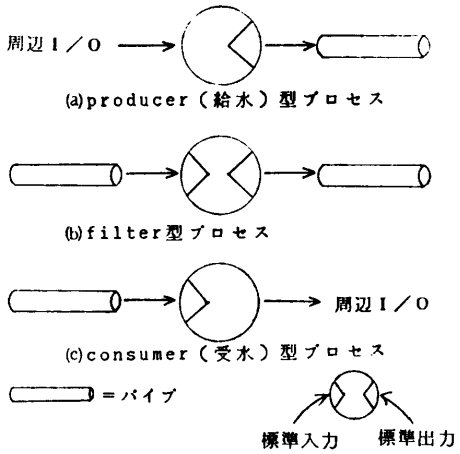

図 8 プロセス基本3型
Fig. 8 Three types of processes using pipes.

表 1 代表的なコマンド展開例
Table 1 Typical command expansions.

種 別	コマンド形式	展開 Remote	展開 Local	コマンド例
整 合	COMMAND	command	| cat	date, ls, who
	COMMAND FILE	command file	| cat	cat, cb, grep, pr sort, uniq, wc
	COMMAND FILE	command file ^^ newcat		ed
不整合	COMMAND FILE	command file		cc, rm
	CP FILE2 FILE1	cat ⟨file2	| cat⟩ file1	cp
	MV FILE2 FILE1	cat ⟨file2	| cat⟩ file1 rm file2	mv
	COMMAND FILE2, FILE1	cat ⟨file2	| cat⟩ temp command temp file1	cmp, comp, diff

4.2 分散処理機能

分散型計算環境のために下記の機能が不可欠である.

①コマンドの遠隔実行

②システム間のファイル転送

③コマンドの並行処理

これらは特に基本的な機能である. それぞれを組み合わせることにより, 種々の有用な応用を実現することも可能である. また, 以上の機能は, 単一システム内における UNIX の機能の, 分散型計算環境への自然な拡張であるため, 利用者はネットワークを意識する必要がない. 上記の処理について, コマンド展開の実例を挙げて説明してゆく.

(1) コマンドの遠隔実行

他系上のコマンドの実行においては, その実行結果を自系に出力することが必要となる. 例えば, 他系上の現在使用中の利用者を表示させるコマンド(who)を実行する場合には, 次のように展開される.

WHO→who | cat

Nshell は, 他系に対して who コマンドの起動を要求し, 自系では cat コマンドを起動する. **図 9** に示すように, 他系上でのコマンドの実行結果は, システム間パイプを通して cat コマンドで自系に表示される.

なお, エディタ (ed) は, 双方向システム間パイプとこれを使用するコマンド (newcat) を用いて, 次のように展開された後, 実行される.

ed FILE 1→ed file 1 ^ ^ newcat

(2) システム間のファイル転送

システム間パイプを用いて, システム間のファイル転送を実現する. 各系では, システム間パイプに対して, cat コマンドがデータの送受を行う. 例えば利用者が, システム 0 (自系) から, システム 2 の file 2 をシステム 1 の file 1 へ転送を行う複写コマンド (cp) を投入したとする. このコマンドは, **図 10** に示すように展開され, 実行される. すなわち,

cp FILE 2 FILE 1→cat 〈file 2 | cat〉 file 1

のようにコマンドの展開を行い, 遠隔対象指向方式に基づいて, それぞれの系へコマンドが送信される. その後, Nshell は, システム間パイプの左右の cat コマンドの起動を, それぞれ, システム 2 とシステム 1 に対して要求する. 起動後, 左側の cat コマンドが file 2 の内容を読み込んで, システム間パイプに書き出す. 右側の cat コマンドはこれを読み込んで, その内容を file 1 へ書き出す. この操作により, ファイルが複写され, 完了情報が自系ディスプレイに表示される.

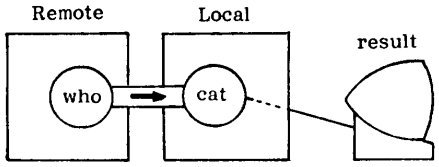

図 9 コマンドの遠隔実行
Fig. 9 Remote command execution.

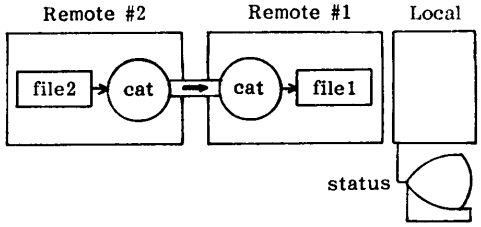

図 10 システム間のファイル転送
Fig. 10 Intersystem file transferring.

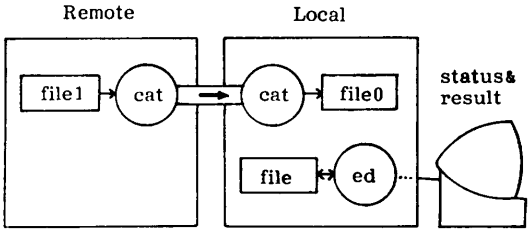

図 11 コマンドの並列処理
Fig. 11 Parallel execution of commands.

(3) コマンドの並行処理

システム間のファイル転送と自系上でのテキスト編集との並行処理を考える. 利用者は, 標準 UNIX と同様に以下のコマンドを連続して投入するだけでよい.

$cp FILE 1 file 0 &

$ed file

ここで, $ はプロンプト, & はバック・グラウンド処理を表す. コマンド投入後, Nshell は上述のコマンド展開を行い, 各系で, それぞれコマンドを起動する (**図 11**). このように UNIX の概念の自然な拡張として, コマンドの並行処理が容易に実現できる.

4.3 応　　用

前述の幾つかの基本機能を利用し, その有用性を確認するために, 応用として下記のものを開発している.

(1) プリント・サーバ, ファイル・サーバ

UNIX の文書作成者ワークベンチ用コマンドを汎用計算機の別の OS (VOS 3) 上に移植し, システム間パイプを用いて (例えば tbl | eqn | NROFF として,

nroff を VOS 3 上で動かす)，清書出力を高品質プリンタに行う．ファイル・サーバとしては，汎用計算機上に UNIX 流のファイル・システムを構築し，動的マウントにより結合して，共用ファイルのサービスを行う．当面は，UNIX System V の全ソース・プログラムの管理に使用する．

(2) 異種ディスプレイ間の画面転送

双方向システム間パイプを用いて，異なる計算機の異種ディスプレイ間で，画面指向の対話を両者に共通の仮想画面とその転送プロトコルにより行う．

(3) 電子メールの転送と管理

文字や図形から成る電子メールを仮想画面として転送することにより，個々のメールを一つの画面として編集・加工することを可能にする．さらに，転送前後のメールを INGRES を用いて統合的に管理する．

表 2 コマンド実行の測定結果
Table 2 Measured command execution times.

処理＼種別	Local [UNIX] (sec)	Local [UNIQNET] (sec)	Remote* [UNIQNET] (sec)
ファイル転送 (4000 bytes)			
cat 〈FILE1 | cat〉 FILE0	2	9～26	85～105**
コマンドの実行			
1) cal 1985	8	12～24	13～ 26
2) who | cat	2	8～17	13～ 25
3) ls-l/| sort | grep unix	6	13～21	70～ 74

* 測定環境 転送速度＝1.2 kbps, パケット長＝400 bytes
転送時間＝転送データ長/転送速度×2 (試用伝送媒体では)

** 転送時間 57～60 (sec)

5. 検　討

試作版 UNIQNET を，利用者インタフェース，異種性やソフトウェアの再利用性・移植性，およびシステムの効率の観点から検討してみる．

(1) 利用者インタフェース

UNIX の概念を自然に拡張したので，異種性の高い環境においても一様な利用者インタフェースを実現できた．実現は，シェル・レベルでコマンドを基盤にして行った．これは，異種性を手軽に解決する有力な方法を与えている．抽象性の高い対象であるコマンドを単位にすれば，別の OS であっても，ある程度の機能を支援できる．しかし，幾つかの問題点も浮き彫りになった．パイプ利用のフィルタ型コマンドが中心となるため，コマンドの解釈や展開などでコマンド・インタプリタが複雑になる．さらに，利用者プロセスの中で他系資源をアクセスするのは，コマンド経由に限られる．コマンドの中にも，そのままでは機能しないものがある．

(2) ソフトウェアの再利用性・移植性

(a) パイプを基盤としたプロセス間通信

標準入出力をインタフェースとする UNIX のプログラミング・スタイルを促進する上から，導入したシステム間パイプによるプロセス間通信は，機能的には極めて満足できる．しかも，これを用いたコマンド・パイプラインにより，ソフトウェアの再利用性を高めながら，分散型計算環境で有用な機能が容易に提供できた．しかし，後述するようにパイプの制限のため，柔軟性・効率などの点で大きな不満が残った．

(b) 仮想画面の導入

仮想端末の概念を画面に拡張した仮想画面は，従来のファイル同様の操作も可能である．特に，コード体系（日本語など）や端末属性が異なる異種型のシステムでは，ソフトウェアの互換性・再利用性を保証する見地から，これらの機能は極めて重要だと思われる．

(c) 拡張部分の移植性

各 UNIX は計算機に依存する独自の改造がなされているものの，System Ⅲ を基盤にしていることもあって，拡張部分は比較的容易に移植ができた．しかし，C言語とそのライブラリの仕様の相違，すなわち，integer の省略時の型，識別子の長さ，システム・コールの有無などが機種により異なり，デバッグ時の困難さを経験した．

(3) 効　率

分散型計算環境のための機能を，UNIX のカーネルに手を加えないで，プロセスと通信機能（パイプとシグナル）を用いて実現した．その結果，**表 2** に示すように，効率は，ある程度試用に耐え得るものの，UNIX が提供する内部通信機能の優劣に強く依存することが明らかになった．既に指摘もあるように，UNIX のパイプにおける柔軟性の欠如が大きな障害になった．すなわち，データのパケット化やポインタ渡し，複数入力パイプの同時待ち合せ，パイプの終了情報の送出，任意のプロセス間でのパイプ敷設などが，UNIX の基本機能として本質的である．例えば，複数パイプの同時待ち合せができないので，**図 12** に示すように，プ

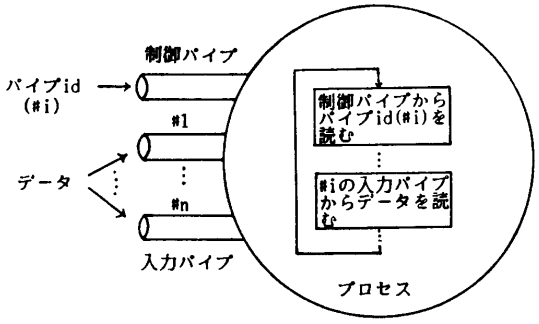

図 12 複数パイプ間の同期
Fig. 12 Synchronization of multiple pipes.

ロセスは制御パイプから制御情報を読み込んで，次に読むべきパイプを決定するという複雑な手法を採らざるを得なかった．これらの要因のため，試作版では，プロセス数の増加，効率の低下，および幾つかの制限を招いたことは否めない．

6. おわりに

シェル・レベルで UNIX の自然な拡張を図ることによって，既存ソフトウェアを再利用しながら，分散型計算環境に必要な機能を比較的容易に実現できた．性能は，当初予想した結果（試用に耐えるが，必ずしも許容はできない）であった．効率の問題は，UNIX 自身の改善(例えば，データの共有メモリ渡しやsocket 等 UNIX の新しいプロセス間通信機能などを用いて)やハードウェアの進歩の相乗効果によって解決できるであろう．UNIX の高い移植性のために，さまざまな方言の出現が懸念されている現在，標準的機能のみを用いて行った我々の接近法は，UNIX（またはこれに代るもの）を分散型計算環境に進化させてゆく重要な1ステップとして有用であると考えている．今後は，シェルを分散環境向けにさらに拡張して，より使いやすい利用者インタフェースを提供してゆきたい．もちろん，分散環境における障害対策や UNIX 自身の分散型計算環境への適応も重要である．

謝辞 最後に，プロトタイプ UNIQNET の設計と製作に携わってきた吉岡和幸，西野浩明の両氏（現在，日本電気(株)）と駒水賢一氏（九州大学）に感謝する．

参考文献

1) 有田，宇津宮：複合計算機の対称型ソフトウェア，情報処理学会第 16 回プログラミング・シンポジウム報告集，pp. 298-311 (1975).
2) Utsumiya, K. et al.: UNIQNET, *Proc. of Pacific Computer Communications Symposium*, pp. 358-360 (1985).
3) 宇津宮：分散型個人用計算環境のための UNIX の自然な拡張，情報処理学会「コンピュータ・システム」シンポジウム論文集，pp. 55-64 (1985).
4) Kernighan, B. W.: The Unix System and Software Reusability, *IEEE Trans. SE*, Vol. SE-10, No. 5, pp. 513-518 (1984).
5) Chambers, F. B. et al.: *Distributed Computing*, p. 327, Academic Press, London (1984).
6) Ritchie, D. M. and Thompson, K.: The Unix Time-Sharing System, *Comm. ACM*, Vol. 17, No. 7, pp. 365-375 (1974).
7) Walker, B. et al.: The LOCUS Distributed Operating System, *Proc. of 9th ACM Symposium on Operating Systems Principles*, pp. 49-70 (1983).
8) Brownbridge, D. R., Marshall, L. and Randell, B.: The Newcastle Connection, *Softw. Pract. Exper.*, Vol. 12, No. 12, pp. 1147-1162 (1982).
9) 砂原，所，中島：S & Tnet 上に構築された分散型オペレーティング・システムの基本機能，情報処理学会オペレーティングシステム研究会資料，24-4, pp. 21-26 (1984).
10) Nowitz, D. A. and Lesk, M. E.: Implementation of a Dial-Up Network of UNIX, *Proc. of COMPCON 80 FDCC*, pp. 483-486 (1980).
11) 駒水ほか：研究室向け LAN のプロトタイプ UNIQNET の通信機能(II)—仮想画面の転送と管理—，第 38 回電気関係学会九支連大論文集，p. 312 (1985).

(昭和 61 年 2 月 17 日受付)
(昭和 61 年 9 月 10 日採録)

宇津宮孝一（正会員）

昭和 20 年生．昭和 43 年九州大学工学部電子工学科卒業．同年より九州大学大型計算機センターに勤務．情報処理教育センター，大学院総合理工学研究科等の助教授を経て，昭和 61 年 11 月より大分大学工学部組織工学科教授．工学博士．オペレーティングシステム，分散型計算環境，日本語文書処理などの研究に従事．電子通信学会，日本ソフトウェア科学会，ACM 各会員．

清水 泰行（正会員）

昭和 36 年生．昭和 59 年九州大学工学部情報工学科卒業．昭和 61 年同大学院総合理工学研究科情報システム学専攻修了．同年富士ゼロックス(株)入社．現在，同社ソフトウェア技術センターに勤務．ローカルエリアネットワークおよびマンマシンインタフェースなどに興味をもつ．電子通信学会会員．