

計算機ネットワークにおける

画像—文章統合型メールシステムの実現

荒川暢也, 高橋薫, 白鳥則郎, 野口正一

(東北大学電気通信研究所)

1. はじめに

最近, 計算機ネットワーク上にメールシステムを構築しようとする動きが活発である。これは, LSI 技術により計算機の小形化, 低価格化およびファクスなどの端末装置のインテリジェント化が可能となったことや, ネットワーク技術の進歩を背景として, 計算機ネットワークに情報処理を行わせ, 事務部門の合理化をめざすOAへの動向を反映している。ところが, 現在実用化されているメールシステム[6][7]はコード化可能な文章情報だけをメールとして取り扱うシステムがほとんどであり, 例外として文章も画像も全て画像と見なして取り扱うシステム[8]が存在するだけである。先に筆者らは文章と画像の両方を, 『領域』という概念を導入し, それぞれの特質を生かして統合化した形式で, 効果的にメールを構成, 蓄積, 転送するシステムを設計した[1]。本論文では, その設計にもとづくシステムの実現について報告する。また, 最近ソフトウェアの保守の問題が深刻化している。従来, ソフトウェアの開発において, 実現後の保守を容易にする有効な方策があまりなかった。そのため, システム実現後にソフトウェアの修正, 変更が生じた場合, それに費される労力, コストは膨大なものとなっている。そこで, このメールシステムのソフトウェア開発においては, 実現後の保守の容易性を目的としたソフトウェアの仕様記述法を用いてドキュメントを整備している。

本論文の構成は次の様になっている。2では, 先に設計した画像—文章統合型メールシステムの概略を説明する。3では, 2で与えられたメールシステムを具体的に実現するための方式設計, 詳細設計, コーディングについて議論を行う。最後に, 実現したメールシステムの実験例を4に掲げる。

2 メールシステムの論理設計

本章では文献[1]で与えられたメールシステムの論理設計を要約して述べる。

2.1 メールシステムの論理構造

計算機網中の各ホストは図1に示すように, そのメールシステムの要素として2つのサブシステムと1つのデータ構造をもち, これらの全体が計算機網メールシステムを構成する。その要素は次の3つである。

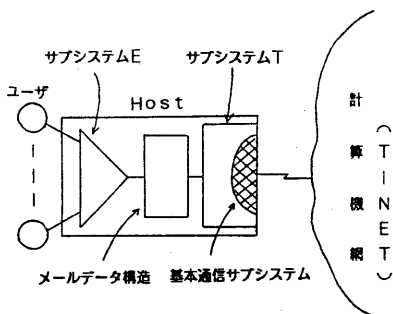


図1 計算機網メールシステムの論理構造

1) メール転送サブシステム(サブシステムT)

2) エンドユーザサービス・サブシステム(サブシステムE)

3) メール・データ構造

2.2 画像—文章統合型メールの構造と構成法

2.2.1 メールの構造

メール1通を図2に示すように『ページ』の集りとし, 各ページを『領域』の集りとして構成する。ここで, 各ページは同一サイズとする。各領域内容は次の(1)あるいは(2)の領域基本情報によって表現される。

<領域基本情報>

(1) コード情報+情報量(縦, 横文字数)

領域はコード化可能な文字列で表現される。情報量はコード情報が縦横何文字で構成されているかを示す。

(2) 画情報+情報量(縦, 横ドット数)

領域は白, 黒を示す2値情報により表現される。情報量は画情報が縦横何ドットで構成されているかを示す。

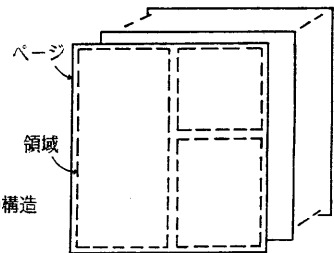


図2 メールの構造

メールの出力デバイスをファクスと想定し, 線密度の異なるファクスがネットワーク内に存在した場合に対処するため, 仮想ファクス(仮想的なメールデバイス)を導入する。その規格仕様を表1に示す。仮想ファクスの導入により, 各ページは図3のように1139×1680の2次元格子として表現され, それを仮想ページと呼ぶ。

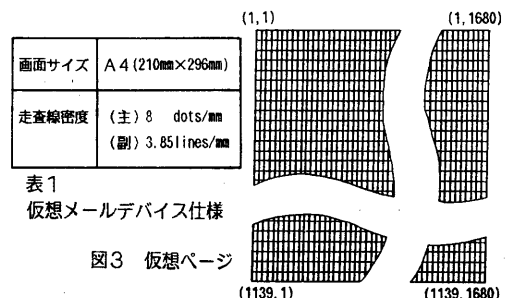
表1
仮想メールデバイス仕様

図3 仮想ページ

2.2.2 メールの構成法

各ページにおける領域の配置は領域基本情報に次のような領域編集条件を与えることにより行う。

1) 文字サイズ(属性がコードの時) 領域内全文字について, それがページ上に配置された時, 縦横何ドットのパターンとして表現されるかを指定する。

2) アドレス 領域を矩形と仮定し、ページ上で領域がどの位置に配置されるかを左肩の座標で与える。

領域編集条件の適用によって領域の位置と内容が確定する。これらの領域に対して2つの演算(+, ×)を定義し、ページ内容を領域の演算結果として構成する。メールはこのようにして構成されたページの全順序集合として構成される。

2.3 メール・データ構造

ユーザによって生成された各領域を論理的な箱の中に収容、管理し、逆にネットワークから配送されたメールの構成領域も同様とする。これらの箱を『メールボックス』と呼び、そのホストに属するユーザに対して個々に用意される。ユーザはキーをシステムに呈示することによってメールボックスへのアクセスが可能になる。そこで、その様な制御のためのデータ構造として『メールディレクトリ』を導入した。メールボックスの構成概略は次のとおりである。

送信部の構成-1) 送信領域管理テーブル、2) 送信領域内容受信部の構成-a) 受信メール管理テーブル、b) 受信領域内容

1) は送信可能領域の識別名(アドレス)、属性、情報量を管理する。a) は受信したメールの識別名、受信日付、発信人名、各ページの領域演算式、メールを構成する全ての領域識別名(アドレス)、属性、情報量、編集条件を全メールについて管理する。2) およびb) は各領域の蓄積内容であり、コード型と画型のそれぞれについて以下のように構成する。

【コード型領域】 内容を行毎に、JISC6220(英数字)及びJISC6226(漢字)に準拠したコードを用いて表現する。

【画型領域】 内容を行毎にMH方式を基本とした圧縮表現として構成する。

2.4 サブシステムEの機能要素

サブシステムEは、メール・ユーザにメールの作成、編集、校正、参照を提供するサブシステムである。その機能要素を次のように定める。

(a) ユーザのアクセス・コントロール (b) コード型領域の作成・編集とメールボックスへの登録 (c) 画型領域の作成とメールボックスへの登録 (d) 送信メールの編集 (e) 送信メール内容の参照、削除 (f) 受信メール内容の参照、削除

2.5 サブシステムTの機能要素

サブシステムTは、ネットワークの伝送制御や論理通信リンクの制御を実現する基本通信サブシステム上に実現され、ユーザレベルのメール転送を提供する。その機能要素は次のように与えられる。

(a) ユーザのアクセスコントロール (b) メール転送 (c) サーバ側ホストに登録されているすべてのユーザ名の参照 (d) 送信メールの記述

2.6 サブシステムEのユーザ・インタフェース

2.4で定めたサブシステムEの機能の提供のために表2のコマンド体系を設計した。なお、テキストのエディット機能のためのコマンドは本設計では与えていない。なぜなら、その機能は各ホストに現存しているものを使えばよいので、本設計ではENTRYCのコマンドによって予めそのようなエディタで作成された領域ファイルをメールボックスに登録する機能だけを与えた。

図4は可能なコマンドシーケンスである。

2.7 サブシステムTのユーザ・インタフェース

2.5で定めたサブシステムTの機能を提供するコマンド体系を表2のように設計した。図5は可能なコマンドシーケンスである。

	コ マ ン ド	機 能
サブシステムT	#LOGON <host-id>	目的ホストとの接続
	#HSP	サブシステムTの呼出し
	#USER <user-id>, <key>	アクセス要求
	#NLIST	メールユーザのリストアップ
	#DESCRIBE <mail-id>	メールの記述
	#MAIL <mail-id> TO <address>	メールの転送
	#BYE	処理の終了
サブシステムE	#DISCONNECT	サブシステムTの終結
	#LOGOFF	目的ホストとの切断
	ACCESS <user-id>, <key>	アクセス要求
	INPUT <file-name>	ファクスから画像領域を入力
	SIZE <file1>, <file2>, <size>	画像領域のサイズ変換
	ENTRYC <file-name>	文章領域をメールボックスへ登録
	ENTRYG <file-name>	画像領域をメールボックスへ登録
	OUTPUTS [, R]	メールボックス中の領域に編集条件、演算を施し出力
	OUTPUTR <mail-id>	メールボックス内の受信メールを出力
	CATALOGS	領域をリストアップ
	CATALOGD	送信メール記述情報をリストアップ
	CATALOGR	受信メールをリストアップ
	ERASES <region-name>	領域を削除
	ERASED <mail-id>	送信メール記述情報を削除
	ERASER <mail-id>	受信メールを削除

表2 メールシステムのコマンド体系

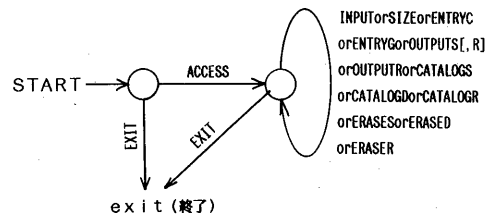


図4 サブシステムEの可能なコマンドシーケンス

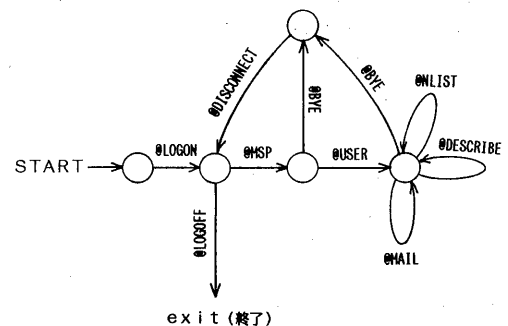


図5 サブシステムTの可能なコマンドシーケンス

3 メールシステムの実現

本章では、前章で述べた論理設計にもとずいて、計算機システム上にメールシステムを実現する方法とソフトウェアの設計（仕様記述とプログラミングなど）について議論する。

3.1 メールシステム実現の方針

計算機上にメールシステムを実現するための労力の大部分はそのソフトウェアの実現に注がれる。ソフトウェアのライフサイクル（図6）において実現とは、外部仕様を受けて方式設計、詳細設計、プログラムの作成、稼働に至る一連の工程である。ところが現在、ソフトウェアを効果的に実現するための決定的な方法論、さらに、最近増大しつつある実現後の保守コストを減少させる有力な手段はない。

ソフトウェア実現の際に要求されることは、

- 1) 論理設計を忠実に反映する。
- 2) 能率的に作業を行なう。
- 3) 実現後の保守を容易にする手段を与える。

ことであり、これらの要求に対し次のような対応策を講じて実現に望んだ。

- a) 論理設計を忠実に反映するモデル化の方法を与え、方式設計を効果的に行なう。
- b) 詳細設計および、実現後の保守に有効であるようにNESD EL-0 [2] による仕様記述を行なう。
- c) 分業やテスト、デバッグ、修正を容易にするため、ソフトウェアのモジュール化を徹底する。

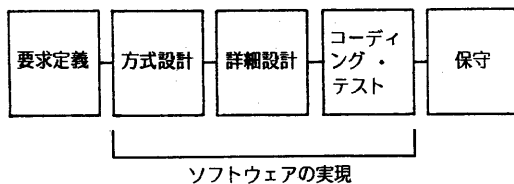


図6 ソフトウェアのライフサイクル

3.2 メールシステムを実現するためのハードウェア構成

東北大通研において、メールシステムを実現するハードウェア構成を図7に示す。ここでファクスの線密度は主走査方向16ドット/mm、副走査方向15.4本/mmで紙面サイズはB4である。ミニコンのメインメモリは32KB、外部記憶（ディスク、ドラム）の容量は3MBである。

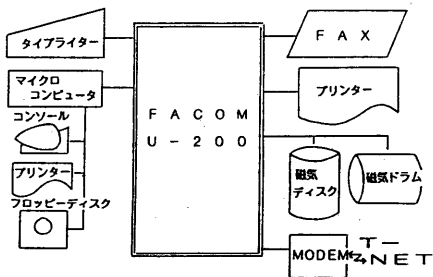


図7 東北大通研のシステム構成

3.3 サブシステムEの実現

3.3.1 サブシステムEの方式設計

方式設計では、2.6で与えられたユーザインタフェース（外部仕様）から実現しようとするソフトウェアのグローバルなモデルを作成する。ここでは、モデル化の方法として、

- a) 各コマンドに対応する機能を実現するソフトウェアモジュールを定義する。
- b) コマンドシーケンスの中にこのソフトウェアモジュールを位置づける。

という方法をとる。このモデルで明らかにされなければならないことは、

- 1) ソフトウェアのユーザインタフェース
- 2) 各モジュールの入出力
- 3) モジュール間の関連

である。そしてこのモデルを以降の作業、および保守の手がかりとなるように仕様記述としてまとめる際には、読者がこのモデルを容易かつ正確にとらえることができるような仕様記述法を用いる必要がある。すなわち、

- 4) 理解の容易性
- 5) 記述の完結性、無矛盾性

が求められる。1)～5)の要求を受けて、仕様記述法NESDEL-0を用いた。

このようにして、ソフトウェアをモデル化し、仕様記述したものが図8である。

モジュール毎のソフトウェア詳細設計の前にプログラム構造を決定する。プログラム構造には先に作成したソフトウェアモデルを反映させ、図9のように決定した。この図において、MAINはユーザのコマンドを解析し、下位のモジュールを統轄するモジュールである。下位のモジュールはMAINより実行権が与えられることにより処理を開始し、処理が終了すると実行権をMAINに返す。

以降は各モジュール毎に開発を進めてゆく。各モジュールは処理手順に着目した詳細化を行なう。

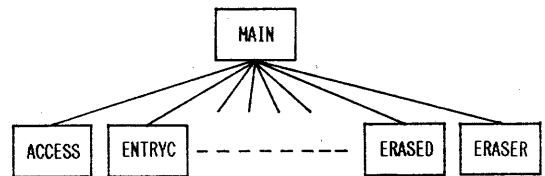


図9 プログラム構造の決定

3.3.2 サブシステムEのモジュール詳細設計

3.3.1で述べたソフトウェアの仕様記述におけるOUTPUTS、INPUTなどのモジュール詳細設計を与える。ここでは特に1つのモジュールについてのみ説明し、他については省略する。

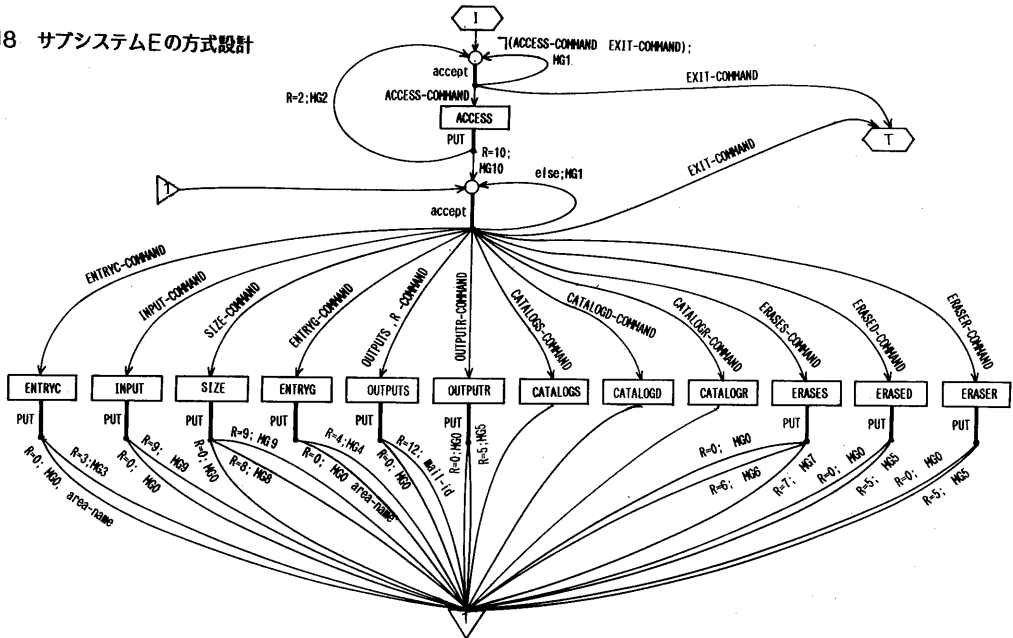
OUTPUTSモジュール

ソフトウェアの機能メールの編集と出力を以下の手順で行なう。

1: ユーザに次のメール編集条件を対話形式で入力させる。

- 1) 論理的領域名（ユーザの与える領域名）
- 2) 領域のページ上のアドレス
- 3) (コード型領域の場合) 文字サイズ
- 4) ページ構成式

図8 サブシステムEの方式設計



```

define
protocol machine( SUBSYSTEM E )
local message (USER): INPUT-(EXIT-COMMAND, ACCESS-COMMAND, ENTRYC-COMMAND,
INPUT-COMMAND, SIZE-COMMAND, ENTRYG-COMMAND,
OUTPUTS-Command, OUTPUTR-Command,
CATALOGS-COMMAND, CATALOG-Command,
CATALOGR-Command, ERASES-COMMAND, ERASE-Command,
ERASER-Command)
OUTPUT=(MG0, MG1, MG2, MG3, MG4, MG5, MG6, MG7, MG8, MG9, MG10)

```

var (R:integer);

meaning

R : return information

local messages:EXIT-Command=<form'EXIT'>

ACCESS-Command=<form'ACCESS'><user-id><key>
ENTRYC-Command=<form'ENTRYC'><character-file-name>
INPUT-Command=<form'INPUT'><file-name>
SIZE-Command=<form'SIZE'><file-name1><file-name2><size>
ENTRYG-Command=<form'ENTRYG'><image-file-name>
OUTPUTS-Command=<form'OUTPUTS'><R'>
OUTPUTR-Command=<form'OUTPUTR'><mail-id>
CATALOGS-Command=<form'CATALOGS'>
CATALOG-Command=<form'CATALOG'>
CATALOGR-Command=<form'CATALOGR'>
ERASES-Command=<form'ERASES'><area-name>
ERASE-Command=<form'ERASE'><mail-id>
ERASER-Command=<form'ERASER'><mail-id>
MG0=no error message
MG1=command incorrect
MG2=access not received
MG3=character file not exist
MG4=image file not exist
MG5=mail not exist
MG6=region not exist
MG7=region exist on mail
MG8=size incorrect
MG9=file name incorrect
MG10=access received

注1 formとはそのコマンドのシンタックスを表すことを意味する。

注2 <>でくくった文字はcharacter strings を意味する。

proc

ACCESS 入力: user-id, user-key
出力: R
user-id およびuser-keyが有効であればR:-10とする。
user-id およびuser-keyが有効でなければR:-2とする。
ENTRYC 入力: character-file-name
出力: R, area-name
character-file-name が有効であれば、その文字ドキュメントファイルを
コード型領域としてメールボックスに登録し、その領域に対しメールボッ
ク内でarea-name をユニークに与え、R:-0とし、さらにユーザにそのarea-na
me を知らせる。 character-file-name が有効でなければ、R:-3とする。
INPUT 入力: file-name
出力: R
file-name が有効であれば、画像ドキュメントをfile-name で示されるフ
ァイルに格納しR:-0とする。 file-name が有効でなければ、R:-9とする。

SIZE 入力: file-name1, file-name2, size

出力: R

file-name1, file-name2, sizeが全て有効ならば、file-name1の画像ドキュ
メントファイルをsizeで示される寸法に変換し、file-name2で示されるファ
イルに格納し、R:-0とする。 sizeが有効でないときはR:-8とする。 size
が有効だが、file-name1またはfile-name2が有効でないときはR:-9とする。

ENTRYG 入力: image-file-name

出力: R, area-name

image-file-name が有効であれば、その画像ドキュメントファイルを送信
用領域としてメールボックスに登録し、その領域に対しメールボックス内
でarea-name をユニークに与え、R:-0とし、さらにユーザにそのarea-name
を知らせる。 image-file-name が有効でなければ、R:-4とする。

OUTPUTS 入力: none

出力: R, mail-id

送信メールを編集し、出力する。 オプションとして「R」が指定され
ると、編集したメールを送信メールとしてメールボックスに登録し、そのメ
ールにメールボックス内でユニークなmail-id を与え、R:-12とし、さら
にユーザにそのmail-id を知らせる。 オプションとして「R」が指定されな
ければ、R:-0とする。

OUTPUTR 入力: mail-id

出力: R

mail-id が有効であればそのmail-id で示されるメールを出力し、R:-0と
する。 mail-id が有効でなければ、R:-5とする。

CATALOGS 入力: none

出力: none

メールボックスに登録されている送信用コード型領域および送信用型領
域のリストを表示する。

CATALOG 入力: none

出力: none

メールボックスに登録されている送信メールのリストを表示する。

CATALOGR 入力: none

出力: none

メールボックスに登録されている受信メールのリストを表示する。

ERASES 入力: area-name

出力: R

area-name が有効で、かつ送信メール上になければ、そのarea-name で
示される送信領域およびarea-name を無効にし、R:-0とする。 area-name が
有効だが、送信メール上にある場合はR:-7とする。 area-name が有効でな
ければR:-6とする。

ERASE 入力: mail-id

出力: R

mail-id が有効であればそのmail-id を無効とし、R:-0とする。 mail-i
d が有効でなければR:-5とする。

ERASER 入力: mail-id

出力: R

mail-id が有効であればそのmail-id で示される受信メールおよびmail-i
d を無効とし、R:-0とする。 mail-id が有効でなければ、R:-5とする。

I

T 終結処理

2: オプション [R] が指定された場合

送信メール管理テーブルにおいてユニークなメール名を与え、送信メール管理テーブルに次の事柄を書き込む。

1) メール名 (ユニークに与えられたメール名) 2) ページ構成式 3) 論理的領域名 (ユーザの与える領域名) 4) (コード型領域の場合) 文字サイズ 5) 領域のページ上のアドレス 6) 実領域名 (ユニークに与えられた領域名) 7) 領域の属性 8) メールボックスにおけるその領域内容の先頭アドレス 9) メールボックスにおけるその領域内容のレコード数 10) 領域の情報量 コード型一行数、横文字数; 画型一行数、横ドット数 2) ~ 5) は1: で入力されたもので、6) ~ 10) は送信領域管理テーブルから、該当する領域の情報をコピーしたものである。

3: 領域の仮想ページ上へのうめこみ

領域を領域演算式にもとずき、仮想ページ上にうめこむ。具体的にはファクス入出力ファイルに領域をうめこむ。

4: 仮想ページの出力

3: でつくられたファクス入出力ファイルの内容を線密度変換し、ファクスに出力させる。3:、4: をメールを構成する全てのページに対して行なう。

このソフトウェアモジュールを実現する際、以下の点が問題となる。

1) 領域演算式の解析方法 2) 仮想ページ上へのうめこみ方法 3) MH符号の復号方法 4) 文字コードの文字パターンへの変換方法 5) ページをファクス出力する際の線密度変換の方法

これらの問題に次のようにして対処した。

1) 領域演算式の解析方法

3: において、仮想ページに領域をうめこむ場合にはまず領域演算式を解析し、領域ごとの2項演算の順序を明らかにしたい。そのために、領域演算式をポーランド記法に変換する。

2) 領域演算式に従った領域のうめこみ方法

領域の仮想ページ上へのうめこみは、メールボックス内の領域内容をファクス入出力ファイルにうめこむことにより行なう。すなわち、ファイル上の演算を行なう。ところが、領域演算式に括弧があれば、領域演算のために、複数のファイルが必要になる場合がある。領域演算はファイル単位に行われるために、外部メモリ容量の小さい計算機システムの場合、容量不足から、そのような演算を実行することは困難である。本研究システムにおいても、外部メモリの容量不足から、そのような演算は暫定的に拒否することにする。

2-1) 画型領域のうめこみ方法

画領域の内容はMH符号の形式でメールボックスに入っている。そこで、画型領域のうめこみは次の手順で行なう。

1] メールボックスから画型領域の1走査線を構成するMH符号列を取り出す。
2] 取り出したMH符号列を復号し、1走査線の画像データを得る。
3] 1走査線の画像データをファクス入出力ファイルにうめこむ。
1] ~ 3] を画型領域を構成する全ての走査線をうめこむまで続ける。

2-2) コード型領域のうめこみ方法

コード型領域の内容はNVTコードの形式でメールボックスに入っている。そこで、コード型領域のうめこみは次の手順で行なう。

1] メールボックスから1文字分のNVTコードを読み出す。
2] そのコードを復号し、32×32のドットパターンをメインメモ

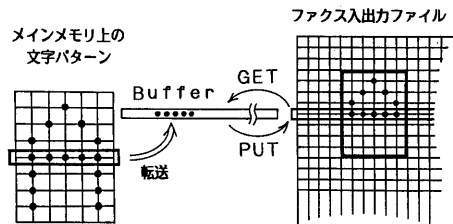


図10 文字パターンのうめこみ

リ上に得る。

3] そのドットパターンを、指定されている文字サイズにSPC法でサイズ変換する。

4] サイズ変換されたドットパターンをファクス入出力ファイルにうめこむ。

1] ~ 4] を、コード型領域を構成する全ての文字をうめこむまで続ける。

うめこみの例として、図10のような9行7列のドットで構成される文字のうめこみについて考える。このドットパターンをうめこむには、文字パターン1行分のデータ毎にうめこまなければならない。そのため、1つのコード型領域をうめこむためには、

(ドットパターンの行数) × (文字数) 回

のファイルへのアクセスが必要となる。これではファイルへのアクセス効率が明らかに悪い。ファイルのアクセスは1レコード毎に行なわれるから、コード型領域1行分の文字を同時にうめこんでゆけばもっと効率をよくすることができる。しかし、文字パターンはメインメモリ上で作成されるため、メインメモリの小さい計算機システムでは、文字パターンのためのメモリを大量に確保することは困難である。ここでは、使用する計算機のメインメモリの制約から、1文字毎のうめこみを行なう。

3) MH符号の復号方法

復号方法は、MH符号のTREEをメインメモリー上に実現し、コード列を1bitずつ読みながらそのTREEをたどり、リーフにたどりつくことによりそのランレングスを得る方法を採用する。このようなTREEによる復号によって変換速度はそのTREEの高さに依存する。

4) 文字コードの文字パターンへの変換方法

変換法として、変換テーブルをファイル上に実現し、NVTコードでファイル上にアクセスし、ドットパターンをメインメモリー上に得る方法を用いた。変換テーブルは、1文字が32×32のドットパターンを1レコードとするファイルである。実現するファイルには、英数字 (JISC6220) と、漢字 (JISC6226の一部) を準備する。

5) ページをファクス出力する際の線密度変換の方法

仮想ページの線密度で作成したファクス入出力ファイルの内容をファクスに送る場合に、次のような線密度変換を行なう。

ファクス入出力ファイルにおける画面の線密度は使用するファクスの線密度より、主走査方向で1/2、副走査方向で1/4である。そこで、主走査方向のデータは各bitを2bitに引き伸ばし、副走査方向には各ラインを4ラインに引き伸ばす。この引き伸ばしは計算機からファクスヘデータを送り出すときに行なう。

3. 4 サブシステムTの実現

サブシステムTは、既の実現されているT-NET [3] の基本通信サブシステム上に、2. 7で定めたコマンドを実現するように構築する。基本通信サブシステムは伝送制御、論理通信リンク制御等の低レベルのプロトコルを実現する。サブシステムTはこの基本通信サブシステム上に構築される機能レベルのメール転送プロトコルを満足しなければならない。メール転送プロトコルおよびサブシステムTの位置づけを明確にするために、プロトコルの階層構造を図11に示す。

サブシステムTは小規模のシステムのため、特に方式設計、詳細設計について本論文で立ち入らないことにする。

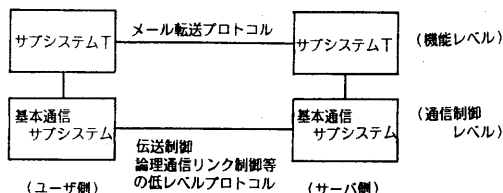


図11 サブシステムTとプロトコル

3. 5 メールデータ構造の実現

2. 3で定めたメールデータ構造を実現するための具体的なファイル構成、レコード構成を決定する。これは、計算機システムに依存するため、ここでは省略する。

3. 6 コーディングについて

コーディングにあたっては、プログラム上ビット処理や周辺機器の制御を行なう必要があるため、アセンブラを使用した。3. 2で与えられたモジュール詳細設計をプログラミングするにあたっては、設計内容の理解の容易性及び保守の容易性を実現するため種々の工夫をしている。たとえば、モジュールをサブモジュールに分割する場合には次の点に留意して行なった。

1) デバックの容易さ プログラムが大きすぎるとデバックを能率よく行なうことが難しくなる。そこでデバックの容易な大きさにサブモジュールをまとめるようにする。

2) サブモジュールの共通部分のサブモジュール化 サブモジュールどうしの間に共通部分のある場合には、その共通部分をサブモジュールとして独立させた方が能率的である。ただし、い

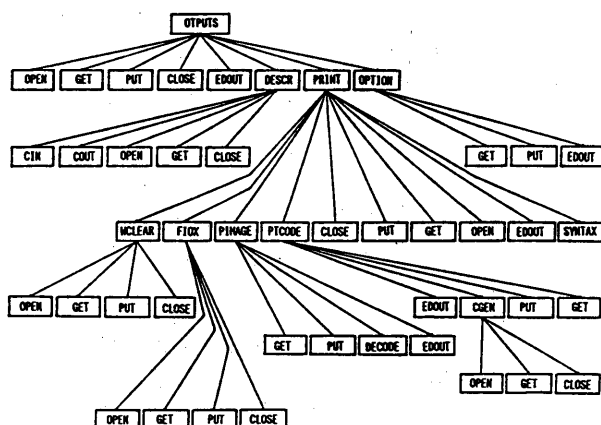


図12 OUTPUTSモジュールの呼出し関係

たずらにサブモジュールを増やしてしまうとサブモジュールの管理が大変になってしまうので注意する。

サブシステムEの実現には約11000steps(67Kbytes)、サブシステムTの実現には約3900steps(20Kbytes) のプログラムを要した。

3. 7 ソフトウェア保守への対策

保守への対策として、プログラム作成者以外の人でも保守が容易に行なえるように、数々のドキュメントを準備した。たとえば、図12のように、プログラムモジュールの呼び出し関係をTREE状にあらわしたドキュメントを準備した。これにより、プログラムの修正、変更の際に、影響の及ぶ範囲を知ることができる。

4 実験

このように実現された本研究のHOSTを用いて、T-NET上でメールの転送実験を行なった結果を示す。

1通のメールを静岡大学のHOSTから本研究のHOSTに転送する。そのメールは1ページからなるメールで、次のような属性、編集条件、領域基本情報から構成されたものである。

- 領域1: 属性(コード型)、情報量(1行×41文字)、文字サイズ(32ドット×32ドット)、アドレス(12, 1)
 - 領域2: 属性(コード型)、情報量(21行×52文字)、文字サイズ(24ドット×32ドット)、アドレス(60, 1)
 - 領域3: 属性(画型)、情報量(346×1600)、アドレス(624, 80)
 - 領域4: 属性(コード型)、情報量(16行×66文字)、文字サイズ(24ドット×24ドット)、アドレス(672, 72)
- ページ構成式: 領域1+領域2+領域3+領域4
(全て+演算)

メール転送に対する静岡大学側のサブシステムTのコマンドのリストをリスト1に示す。

```
?@LOGON NOGUCHI (1)
LOGGED ON
?@MSP (2)
MSP OPENED
INPUT MSP COMMAND
?@USER K.OCHIMIZU OF SHIZUOKA-UNIV. (3)
COMMAND RECEIVED CORRECTLY
INPUT MSP COMMAND
?@DESCRIBE MAIL1 (4)
MAIL DESCRIPTION TABLE RECEIVED
INPUT MSP COMMAND
?@MAIL MAIL1 TO K.TAKAHASHI (5)
MAIL TRANSFER STARTED
AREA TRANSFER COMPLETED
AREA TRANSFER COMPLETED
AREA TRANSFER COMPLETED
AREA TRANSFER COMPLETED
MAIL TRANSFER COMPLETED
INPUT MSP COMMAND
?@BYE (6)
BYED
?@DISCONNECT (7)
DISCONNECTED
?@LOGOFF (8)
LOGGED OFF
```

リスト1 サブシステムT使用例

- (1) メールの宛先人が所在するホスト（東北大通研）に物理リンクを接続する。
- (2) メール転送を行なうためにサブシステムTを呼び出す。このコマンドにより、サブシステムT間に論理リンクが開設され、サブシステムTが起動される。
- (3) ユーザ側サブシステムTに対してはアクセス権の獲得およびメールボックスのオープンを申請する。サーバ側サブシステムTに対しては、後にメール内容を転送した時のメール発信人を示すために用いられる。
- (4) メールの記述を行なう。この例ではメールの記述はリアルタイムには行なわず、予めローカルに行ない、送信管理テーブルに登録しておいてから一括してサーバ側サブシステム

- Tに送っている。
- (5) MAIL1で示されるメールをK. TAKAHASHI で示されるユーザに転送する。転送は各領域単位に送られる。
- (6) ユーザがユーザ側サブシステムT、サーバ側サブシステムTの利用の終了をつける。
- (7) 論理リンクをクローズし、サブシステムTを終結させる。
- (8) 物理リンクを切断する。

本研究室で受信したこのメールをサブシステムEのOUTPUTコマンドで出力したものが図13である。各領域データは目的地に正しく送られ、メールとして組み立てられ出力されている。このような、メールの作成、蓄積、転送の実証実験から、この画像-文字統合型メールシステムの設計の正しさが示された。

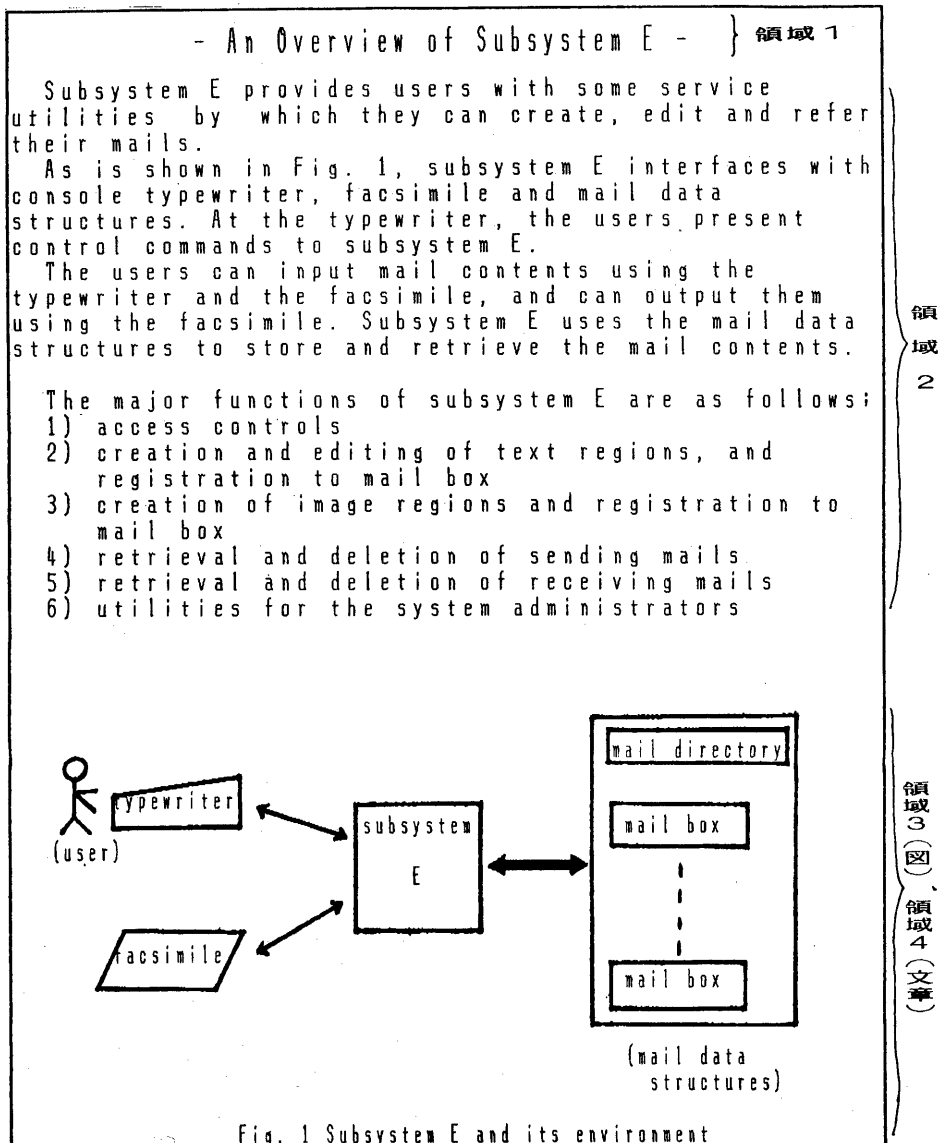


図13 実験メール

この画面をファクス (G-3, 8dots/mm, 3.85lines/mm) で扱うと、データ量は 214.2KBとなる。この画面を全てMH符号化すると、57.8KBであり圧縮率は 0.27 となる。サブシステムEを用いて各領域 (AREA1 ~ AREA4) に分割し、それぞれの領域を属性によって圧縮した場合には、トータルのデータ量は 7.189KBで、圧縮率は 0.033である。したがって効率の良い転送がなされていることが示される。

次に、本研究室に転送された図13のメールと全く同じものを本研究室のサブシステムEで編集、出力する実験を行なった。これはサブシステムEをローカルなエディタとして使用したものである。4つの領域内容は既にメールボックスに入っており、システムからそれらの領域に対しユニークにAREA1 ~ 4の領域名が与えられている。このときOUTPUTSコマンドを入力すると、リスト2のように対話形式でメールが編集される。

*OUTPUTS

- DESCRIPTION OF REGIONS AND EDITING CONDITIONS -

REGION NAME : R1=AREA1, R2=AREA2, R3=AREA3, R4=AREA4 (1)

CONDITION : ADDRESS OF R1 IS (12,1) (2)

SIZE OF CHARACTER IS (32*32) (3)

ADDRESS OF R2 IS (60,1) (2)

SIZE OF CHARACTER IS (24*32) (3)

ADDRESS OF R3 IS (624,80) (4)

ADDRESS OF R4 IS (672,72) (2)

SIZE OF CHARACTER IS (24*24) (3)

DESCRIPTION OF PAGES -

PAGE 1 EXPRESSION = R1+R2+R3+R4 (5)

PAGE 2 EXPRESSION = (6)

リスト2 サブシステムE使用例

- (1) システムから与えられたユニークな領域名 (AREA1 ~ 4) に対し、メール記述を行なうための仮想的な領域名 (R1~R4) を与える。
- (2) コード型領域 (CODE) のアドレスを指定する。
(縦ドットアドレス, 横ドットアドレス)
- (3) コード型領域 (CODE) の文字サイズを指定する。
(縦ドットサイズ*横ドットサイズ)
- (4) 画型領域 (GRAPHIC) のアドレスを指定する。
(縦ドットアドレス, 横ドットアドレス)
- (5) 第1ページのページ構成式を与える。

メールの編集が終了すると、そのメールがファクスから出力される。

5 むすび

本論文では、筆者らが先に論理設計したメールシステムを具体的に計算機上に実現した。はじめに、インプリメンテーションの方式設計を与え、次に詳細設計、コーディングを行なった。方式設計では、変更および保守の容易性を実現するため、仕様記述形式NESDEL-0を用いて実現システムの仕様を与えた。また、詳細設計では、方式設計で与えられた各機能モジュールを実現するための問題点を整理し、これらを解決するための効果的なアルゴリズムを与えた。さらに、プログラムの段階では、プログラムモジュール間の呼び出し関係図を作成することにより、プログラム保守の容易性をはかった。ただし、本メールシステムの設計、実現において、その性能についての厳しい要求を課さなかった。そのため、処理時間、処理効率の面で問題が残されている。

<謝辞>

メールシステムの開発および転送実験に多大なご協力を頂いた、静岡大学落水浩一郎助教授、同平田明良君に感謝致します。

<参考文献>

- [1] 高橋薫, 白鳥則郎, 野口正一: "計算機網における画像-文字統合型メールシステムの設計", 信学技法 EC82-45 (Oct. 1982)
- [2] 高橋薫, 白鳥則郎, 野口正一: "ネットワークソフトウェアの設計法", 情報処理学会ソフトウェア工学研資料(1983)
- [3] 野口正一: "公衆回線による計算機ネットワークの研究", 文部省特定研究「情報網システムに関する基礎研究」報告書(Sept. 1978)
- [4] 高橋薫, 白鳥則郎, 野口正一: "Computer Mail Systemの設計", T-NET内部資料(Aug. 1982)
- [5] 荒川暢也: "計算機ネットワークにおけるメールシステムに関する研究", 東北大学昭和57年度卒業論文
- [6] T.H. Myer: "Future message system design: lessons from the HERMES experience", COMPCON Fall (Sept. 1980)
- [7] 鈴木他: "インハウス・ネットワークに於けるエレクトリックメール・システム (ELMS) について", 情報処理分散処理システム研究会11-5 (1981)
- [8] P.T. Kirstein and S. Yilmag: "Facsimile transmission in message processing system with data management", ICCM (1978)
- [9] W. Horak and W. Woborschi: "TEXTFAX-principle for new tools in the office of the future", MCC, AFIPS (1979)