FEBRUARY 1983

HEWLETT-PACKARD JOURNAL

Electronic Mail for the Interactive Office

Here's how electronic mail is implemented on the HP 3000 Computer System. HPMAIL lets users who aren't familiar with computer technology exchange messages effectively within their organization.

by Ian J. Fuller

HPMAIL IS AN ELECTRONIC MAIL SYSTEM that operates on the Hewlett-Packard 3000 Computer System. It is designed to enable users who may not be familiar with computers and their associated technology to interchange information effectively throughout their organization. With HPMAIL, messages, documents, business charts and graphs, and HP 3000 system files can be exchanged both locally and remotely.

HPMAIL operates concurrently with other data-processing and office information-processing activities. It operates from any terminal that can be connected to an HP 3000. That includes the HP 2626W Word Processing Station, the compact HP 2382A Office Terminal, the HP 125 Personal Office Computer, and any of the large range of other HP terminals and desktop computers with data-communications connections to HP 3000s.

Using standard Hewlett-Packard Distributed Systems Network capabilities, HPMAIL manages all message routing and data communications among multiple HP 3000 Systems. A store-and-forward feature facilitates transmission through the nodes of a multiple-system network, providing flexible network paths through existing data-communications connections. These may be over dial-up, leased, direct-connect or public data network (X.25 and X.21) lines. Users need only specify the name of the recipient. Actual routing paths are invisible to the user. A general delivery capability routes messages to locations where they can be distributed manually.

Each user has a set of software tools that map onto the traditional elements of the desk. An electronic in tray keeps track of incoming messages, while the out tray organizes mail being sent. Users monitor the progress of messages in the pending tray. Senders can not only verify that their messages have arrived, but also that they have been read. A work area provides for composing and editing longer communications, and for assembling packages of information files, including graphics. A file cabinet stores messages and documents. A distribution directory lets the user construct, use and store standard distribution lists. An administration area provides the means to tailor the HPMAIL environment with passwords, autoforward instructions, autoanswer messages, and the choice of a person to handle mail on the user's behalf.

Simple commands perform everyday mail functions. Movement between HPMAIL areas is by function keys, automatically labeled on the screen when that ability is supported by the particular HP terminal.

To reduce the on-line system storage traditionally required by computer-based message systems, HPMAIL uses an HP 3000 IMAGE data base for document storage and local distribution. Documents are stored only once on each system in a network with pointers for each intended recipient.

HPMAIL Functions

The basic facilities required of any electronic mail system are to accept mail, address users, store mail, sort outgoing mail, move the mail, sort incoming mail, deliver mail to recipients, and notify recipients. HPMAIL performs all these functions. It also tracks the progress of mail and provides a flexible and powerful user interface.

HPMAIL can transmit any information that can be stored in the HP 3000 system. This can range from a few lines of simple text to a complex report containing, perhaps, reports produced by a word processor such as HPWORD, graphics produced by DSG/3000 or HPDRAW, and even HP 3000 program and data files. We decided to adopt as flexible a structure for these items as possible, enabling the user to organize them into packages of logically related items. At the same time, we realized that the majority of HPMAIL messages would be simple text and made it very simple for this type of message to be accepted into the system.

HPMAIL can be implemented as a multicomputer system, but it is important in the office environment that users be shielded from the details of the configuration of their particular network. To this end we implemented an addressing scheme. HPMAIL users are registered with the system in such a way that senders of messages do not need to know whether recipients are located on their computer or on one many miles away, connected through a complex network. It also allows the administrator of an HPMAIL network to reconfigure it without users having to change the way in which they address other users.

The system must be able to store mail before transmission, on intermediate computers in a store-and-forward network, and on the behalf of users. Because of the potentially large space requirements for messages on a system, HPMAIL has adopted the strategy of sharing information wherever possible. Thus, when a message is delivered to several users on the same computer, only one copy is stored and pointers are set up linking the message to each user's in tray.

Mail in transit must be sorted into queues for transmission to the required destination and for distribution at the destination computer, a process analogous to the operation of a sorting office in a manual mail system. This can be a

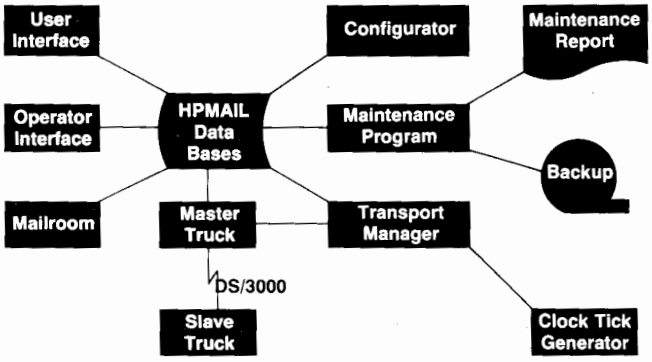


Fig. 1. *Major software components of HPMAIL.*

time-consuming process, involving the analysis of large distribution lists, and so we decided to have this function performed by a background process called the mailroom.

Mail must be transported between computers as economically as possible. HPMAIL uses the program-to-program (PTOP) communication facility provided by DS/3000 for this task. These potentially lengthy and error-prone operations are handled by autonomous processes known as the transport manager and the master and slave trucks.

Once arrived at its destination computer, mail must be delivered to its recipients and the recipients must be notified of their new mail if possible. This work is also handled by the mailroom program.

For users to have confidence in the operation of the mail system they must be able to track the progress of messages that they have sent. HPMAIL provides five acknowledgment levels that users can put on a message and the transport system keeps users informed of the progress of their messages according to the acknowledgment level that they have set.

The HPMAIL user interface provides users with powerful facilities to create and store messages on their electronic desk, making HPMAIL far more than a simple message system. Examples of the facilities provided are the work area in which items may be constructed before incorporating them in messages, the filing cabinet, in which users may store messages for future retrieval in folders they have named, and the distribution list area, where users can keep distribution lists of other users with whom they correspond for easier compilation of messages. When we designed the HPMAIL user interface we placed considerable stress on ease of use for the nontechnical user and compatibility with any terminal that can log on to the HP 3000. This lets users access their mail even when they are away from their office if they have a portable terminal and access to a telephone.

Implementation of HPMAIL

One of our principal design objectives in implementing HPMAIL was that the product should be easy to maintain, test, debug, and update. To this end we decided upon a layer concept for the software, with each layer communicating via rigidly defined interfaces. This allowed development of the transport mechanism to proceed relatively undisturbed while the details of user interface design were refined through a series of tests, prototypes, and evaluations. The task of integrating these components, often a major headache in software projects, proved to be a relatively simple task with HPMAIL and the software has proved to be very robust in use.

Fig. 1 shows the major software components of the product. The user interface program is responsible for all the user interaction in the product and provides the user with the electronic desk described previously. The mailroom is a background job streamed by the system operator and is charged with the sorting and delivery of local mail, that is, mail that is destined for users on the same computer. The transport manager is responsible for scheduling the communication of messages between HPMAIL computers according to the current message load and the availability schedule set up by the HPMAIL system administrator.

The master truck is controlled by the transport manager and is responsible for the setup and control of the DS/3000 connection with a remote computer where it uses the slave truck program to receive the mail and put it onto the remote data base. The clock tick generator runs as a son process of the transport manager and provides it with a tick every fifteen minutes to cause the transport manager to reschedule its transmission priorities as necessary.

The configurator is a VPLUS/3000 application program used by the HPMAIL system administrator to build and configure the HPMAIL data-bases. Facilities are provided so that the administrator does not need to have any detailed knowledge of IMAGE/3000 to control HPMAIL.

The operator interface is a set of functions that enable the system operator to control HPMAIL and determine its status. Examples of operator commands provided are MAILON and MAILOFF to enable and disable the system, MAILSTATUS to provide a status display for the system, MAILSHOWNODE to display the number of messages awaiting transmission to remote computers, and MAILMAINT to start the maintenance program.

The maintenance program is run periodically to provide verification of the intactness of the HPMAIL data bases, to delete outdated mail items, and to generate statistics, which are formatted and printed by the report program. Regular running of the maintenance program is a vital part of maintaining a reliable HPMAIL system.

Development Techniques

The programs described above were developed simultaneously over a period of eighteen months. For this work to proceed reliably a set of techniques was derived before detailed design began.

Three techniques we adopted in the development of the product deserve mention here as being important to the way in which the product evolved. These are the use of mail procedures to access the data bases, IPC or pipe files to communicate between the components of the product, and the standardized tracing and error reporting mechanism used throughout the product.

Mail Procedures. The HPMAIL software is built around two IMAGE/3000 data bases, called LOCAL and GLOBAL, which are discussed in the next section. The application programs that access the data bases, for example the user interface, mailroom, transport manager and trucks, do so through a set of intrinsics (routines) called mail procedures. These mail procedures are in a segmented library and so are

accessible to all the programs. They perform the low-level data base access functions and insulate the programs that call them from the details of the data base structure. In fact, the transport programs do not have a single IMAGE call in them; all interaction is done through mail procedures. Examples of the mail procedures include MOPEN to open the mail data bases, MCREATE to create a new item, MEXPLODE to return information about the contents of an item, for example the messages in a user's in tray, MATTACH to attach one item to another, for example, to attach a new message to a user's in tray when delivering it, MDELETE to delete an item, and MCLOSE to close the mail data bases.

Other mail procedures were implemented to perform common functions required by different modules, for example, comparing two names. In all, about thirty mail procedures exist in the HPMAIL segmented library.

These procedures were designed, tested, and made available before any coding was done on the programs that would use them, thereby providing a firm base upon which development could proceed.

IPC Files. Besides using the two IMAGE data bases, the various components of HPMAIL communicate with each other using the interprocess communication (IPC) or pipe facility provided by the HP 3000's MPE-IV operating system. This method of communication has the advantages of being very well defined and straightforward. It is reliable and provides an interrupt mechanism whereby a program can wait for a message on its pipe telling it what to do next. Once again, this convention aided the product implementation. Modules were tested using simple test harnesses before attempting to integrate them.

Three IPC files are defined in HPMAIL. MRIPCIN is the mailroom input pipe, which is written to by the user interface when a user MAILs a message, instructing the mailroom to collect it from the user's out tray and deal with it. It is also written to by the slave truck when it receives mail from a remote computer bound for the local computer, alerting the mailroom to deliver it.

TMIPCIN is the transport manager input pipe, which is written to by the mailroom to inform it about new mail bound for remote computers which it should attempt to send, and by the clock tick process, which runs as a son of the transport manager and sends it a message on this pipe every fifteen minutes to cause it to reschedule itself to take account of any change in the availability of routes out of the local computer.

MTIPCIN is the master truck input pipe. The transport manager sends instructions, such as "open a DS line to computer X," to a master truck, which attempts to perform this operation and reports back to the transport manager on its input pipe the success or failure of the operation. The transport manager can support up to eight master trucks simultaneously and each has a separate input pipe, referred to as MTIPCIN0, MTIPCIN1, etc.

Tracing and Error Reporting. Since HPMAIL consists of a number of intercommunicating programs and each program consists of a large number of modules, each separately compiled, it is important that status information about the success of each operation is returned to the caller. It is also important to have a trace facility built into the product so that errors can be traced and diagnosed with minimum trouble.

In HPMAIL, a single data structure called the mail common area is used to communicate shared data such as completion status, file numbers, and user information between the various procedures in a program and with the mail procedures. This is very much in line with products such as VPLUS/3000 or IMAGE/3000.

Each procedure in the product (and there are over 300) takes the mail common area as its first, and sometimes only, parameter. The convention was established that all the procedures call a mail procedure called MTRACE immediately upon entry and before exit. MTRACE registers the procedure's identity in the mail common area and can also be made to write a log message to a file if required.

Error reporting is centralized in another mail procedure, MERROR, which interprets the status words in the mail common area and can produce a customized error message from the message catalog for each error, together with as much additional information as can be deduced. Since all the trace messages are derived from the message catalog, they can, if necessary, be translated to other languages.

The tracing level of these procedures can be varied by setting a job control word (JCW) or by passing a run-time parameter (PARM) to the program. Thus, when debugging a program, the engineers can easily get full trace information and errors can be located more easily. This level of control was particularly important in testing the transport programs, where it was not practical to use the MPE debug facility. These tracing facilities remain in the released product, but cause little performance penalty since they are normally disabled.

HPMAIL Data Bases

HPMAIL uses two IMAGE/3000 data bases to contain the user and network directory information and the users' elec-

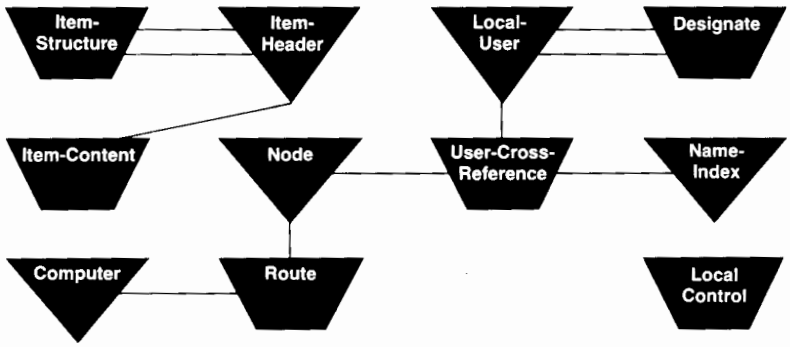


Fig. 2. *The HPMAIL LOCAL data base contains local directory information about users, network directory information, and mail items.*

tronic desks, all maintained securely and reliably.

All the HPMAIL programs on a single computer work with the same data base set. Items that are simultaneously owned by more than one user, for example a message delivered to two in trays, are held physically only once per system, with pointers to assert individual ownership.

The HPMAIL GLOBAL data base is a directory of all users known to this system. The LOCAL data base contains local directory information about the users supported there, network directory information to control intercomputer transmission, and mail items. Mail items can be in trays, messages, queues of messages, individual text items, distribution lists, or MPE files imported onto the data base—in fact, nearly anything an HP 3000 can store.

The HPMAIL data bases perform three functions: directory, storage, and item composition. The directory function enables HPMAIL to identify users when they sign on and determine those items that they can access. The directory also identifies users on mail distribution lists and enables user names as entered to be expanded to the full canonical form of name, location and sublocation. The network directory functions enable HPMAIL to determine which computers to contact to send mail to a given location, how and when to contact the computers, and what telephone number or public data network (PDN) node name to specify.

The storage functions enable HPMAIL to hold everything in the IMAGE data bases so that the information is always available. Thus HPMAIL will take an HPWORD document and import it onto the LOCAL data base so that at some later time the transport system will be able to send it to another computer without requiring access to the sending user's MPE group, or having to worry if the user deleted it.

The composition functions enable HPMAIL to handle hybrid message types, to have the same message in several in trays or filing cabinet folders at the same time, and to operate queues of messages destined for different locations in the network. This leads to the important concept in HPMAIL that the simplest message contains two parts: a distribution list and some content (usually text). The message itself is a single item that contains these two parts. This split enables the transport mechanism to refer to the message as a whole, while the mailroom can access the distribution list, its primary concern, without having to scan the text. The end users also derive the benefit that they do not have to read the message distribution list but can go straight to the text.

HPMAIL uses the IMAGE/3000 Data Base Management System for many reasons. Almost the least important is that it is a data base management system. The main reason is that it is solid and well-established and provides a firm foundation for a reliable product.

It also offers its own space management, allocating new space as required and handling direct access problems transparently. Because of the hybrid nature of most HPMAIL items, the chain management system of IMAGE is very important. It handles pointer chains very easily. IMAGE uses shared data storage and shared buffering, adding to efficiency and performance. It has a good locking mechanism to prevent unreliable operation caused by multiple updates of the same item. Finally, an IMAGE data base can be dumped for security purposes very much faster than many MPE files.

The main drawback to using IMAGE is that it is not designed to hold nonstructured items such as MPE files, and getting them in and out of the data base (importing and exporting) is a fairly lengthy process. We estimate, however, that the time required to implement the project would have nearly doubled if the facilities of a system like IMAGE had not been available.

Data Base Descriptions

Figs. 2 and 3 show the schemas (definitions) for the two data bases. Two examples of the way HPMAIL uses the data bases in the execution of common functions are item storage and name searching.

The most important data sets to the understanding of HPMAIL's item storage are the ITEM-HEADER, ITEM-STRUCTURE and ITEM-CONTENT data sets on the LOCAL data base. Items in HPMAIL are considered to be either basic, meaning that they can be represented by a single MPE file (for example a distribution list or an HPWORD document), or composite, which have no MPE file form as such but are composed of a number of basic items. An in tray is an example of a composite item, consisting of a header linking it to the messages in that particular in tray.

Every mail item, basic or composite, has a type (e.g., user folder, message, text), a creator, and a subject. This information is kept in its item header. The ITEM-STRUCTURE data set maintains the links between items that make up composites. For example, a message consists of a message header linked by two item structure records to the headers of its distribution list and content. The content of a basic item, the MPE file content, is held in the ITEM-CONTENT data set, linked by IMAGE pointers to its item header.

In HPMAIL, every item is linked to at least one other, except for the root item, which we call item zero. If an item becomes detached from all its parents, which happens, for example, when a message is deleted from the in trays of all the users who received it, it becomes an orphan item and will be physically deleted from the data base by the maintenance program.

Fig. 4 shows the HPMAIL object hierarchy in more detail. Although superficially complex, it is in fact quite efficient and lends itself very well to processing by IMAGE via the mail procedures.

Searching for Names

Every time a user enters a name into the user interface program, whether it is the user's own name entered during the HPMAIL sign-on, or names on a distribution list,

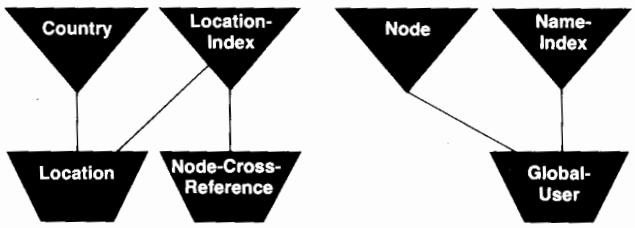


Fig. 3. *The HPMAIL GLOBAL data base is a directory of all users known to the particular system.*

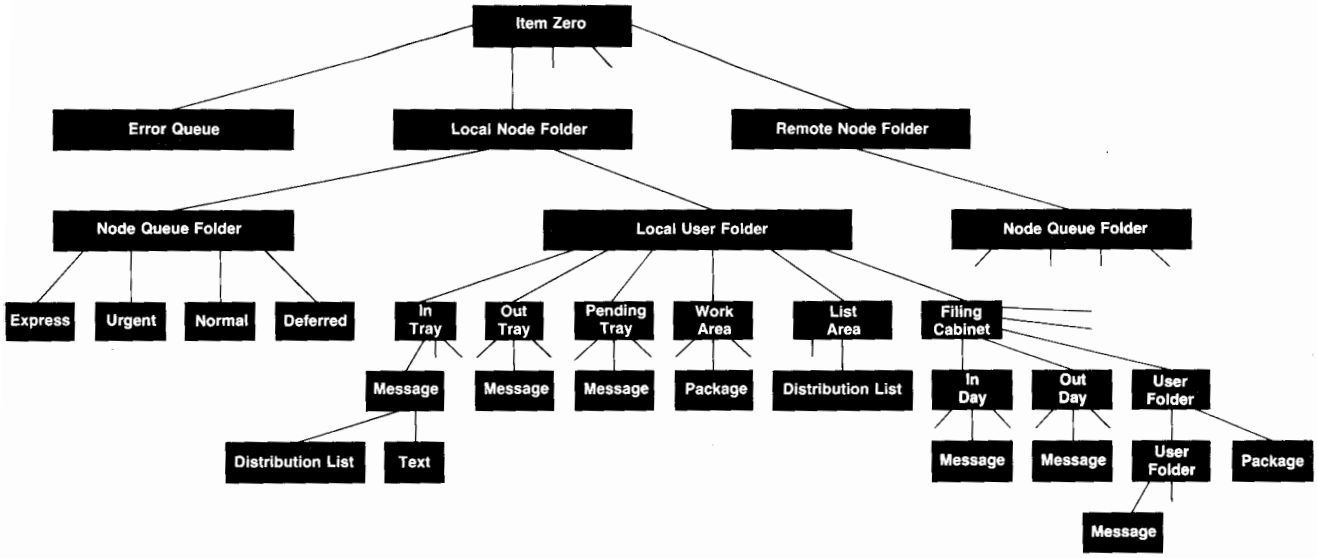


Fig. 4. *HPMAIL object hierarchy.*

HPMAIL searches its directory for that name or a similar match. Names in HPMAIL are held in what we call canonical form, which is:

<SURNAME>,<FIRST NAME> <MIDDLE NAME>/
<LOCATION>/<SUBLOCATION>

where the surname, first name, and middle name add up to no more than 36 characters, the location is up to six characters long, and the sublocation is two characters long. Canonical names are all uppercase, converted by means of an internal translation table to take care of national character sets. For example, the canonical form of user Ian John Fuller, location HP1600, sublocation 01 is:

FULLER,IAN JOHN/HP1600/01

HPMAIL does not insist that the user type in the full name, just enough to identify it uniquely on the data base. Thus if there are no other users with the surname FULLER defined on the data base, FULLER will be sufficient. If any doubt exists, the user interface displays those available and asks the user to choose one. Of course, user names must be unique to a mail node (location+sublocation).

When HPMAIL searches the data base for a name, it does so by first making a name probe from the surname supplied. The name probe consists of the first letter of the surname, followed by the next three nonrepeating consonants. For example, the surname FULLER yields the name probe FLR. This probe is taken as the key to a search of the NAME-INDEX master data set, which will give a chain of names that have this probe. The user interface scans down this chain comparing each with the canonical form of the input name. There may be several degrees of match: all components agree, first names differ, locations differ, etc. If a perfect, nonambiguous match occurs, then the user is not troubled further. Otherwise the user is given a choice of those available. The use of the name probe means that a slight misspelling of a user's name can sometimes but not always yield a choice that reveals the correct spelling of the name the user was looking for. For example, if the user types the surname COALMAN instead of COLEMAN, then the program will be able to give a choice of the correct name since both names result in the name probe of CLMN.

In the case of the user sign-on process, the LOCAL data base is searched for the name, since it holds the name information for the users supported on that computer. For the construction of distribution lists, the GLOBAL data base is searched since it contains the directory of all the users known to that computer, but not necessarily supported on it. The user has no need to know where another is supported; this is sorted out automatically. The name searches are done in a similar way on both data bases by mail procedures.

User Interface

The HPMAIL user interface provides the user with an environment designed to mirror the concepts familiar to office users. As mentioned earlier, the user is provided with an in tray into which incoming messages are delivered by the mailroom, an out tray in which messages the user sends are assembled and from which they are collected by the mailroom for transmission and delivery, a pending tray which holds messages awaiting acknowledgments (the user can find the acknowledgment status of any message in the pending tray by reading its distribution list), a work area which allows the user to create items and assemble them into packages before incorporating them into messages, a distribution list area which holds lists of users with whom the user may wish to communicate, a filing cabinet in which the user can file messages for future reference, and the administration area which gives users extra commands to do such utility functions as defining or changing their passwords and naming persons who may work on their behalf.

The command set of HPMAIL is designed to reflect the

terminology used in offices. Examples of HPMAIL user interface commands are:

READ	to display an item on the user's terminal
PRINT	to print an item on the system line printer
SEND	to start to send a message
MAIL	to commit a message for transmission
FILE	to copy an item to a folder in the filing cabinet
REPLY	to send a reply to a message in the in tray.

The function keys of Hewlett-Packard terminals are used if possible to give the user an easy method of using the product. Function keys **f1** to **f7** select entry to each of the areas listed above. **f8** allows the user to leave HPMAIL.

The prompt displayed by HPMAIL is another aid to the user. This changes according to the area. The in tray prompt is IN TRAY>, the filing cabinet prompt is CABINET>, etc. When the user has OPENed a folder the prompt changes to the first word of the folder's label, so a folder with the label User Group Presentation will, when OPENed, give the prompt USER>.

Fig. 5 shows an example of HPMAIL's main menu, which is displayed as soon as the user signs on. Following that, a typical user presses function key **f1**, selecting the in tray to read any new messages. These messages are summarized in a list as shown in Fig. 6. Note also how the prompt has changed from HPMAIL> on the main menu to IN TRAY> to indicate the change of area.

Typing the command READ 1 after the IN TRAY> prompt causes the first message in the in tray to be displayed on the user's terminal. Fig. 7 shows an example of the format of a message displayed by HPMAIL.

The user interface also offers an extensive HELP facility which is organized on three levels. Typing the HELP command by itself gives general help about the area the user is in. For example, HELP in the filing cabinet gives the new user some idea of the purpose of this area and suggests some likely commands. The command HELP followed by another command name gives the user specific help about that command as it is used in that area. For example, typing HELP MAIL while composing a message explains the meaning of the command and how to use it. Finally, typing a question mark (?) in any area gives the user a brief list of the commands available.

HPMAIL was designed to be translated easily into other languages. The translator doesn't need knowledge of the internal workings of the product or access to the source code. To this end, all messages produced by the product come from a central message catalog (which is currently about 4,000 lines long) while the HELP displays come from another catalog which is even longer.

All user input except the content of messages is parsed by common parsing routines and commands are recognized by reference to a dictionary. The development team wrote a special utility to build this dictionary, which goes under the name of Webster. It is available to Hewlett-Packard software centers worldwide to help them adapt the product to their local language. Synonyms and common abbreviations for commands can be built into the dictionary as required for each language.

Local character sets are handled by a translation table as part of the dictionary so that users may, in their local versions, use commands containing their local characters. One limitation to the use of local character sets in HPMAIL, however, is that user names, locations and sublocations have to be in the standard USA ASCII code. This is because an HPMAIL network can span countries, not all of which may use the same conventions regarding extensions to the character set.

User Interface Structure

The HPMAIL user interface program is a direct analog of the data structures it supports in the LOCAL data base. It is a direct result of the maxim, "Design the data structures correctly and the applications programs will almost design themselves." The program is a tree structure, as shown in Fig. 8, which follows the object structure quite closely.

The outer block of the user interface controls all the other

```
HPMAIL                                                    1325 02/07/83

Ian Fuller                                          Location HP1600/01

You have 3 new messages.

Select     1   IN TRAY - Read your incoming mail
           2   OUT TRAY - Send messages
           3   PENDING TRAY - Track messages you have sent
           4   WORK AREA - Create new work
           5   DISTRIBUTION LIST AREA - Create new lists
           6   FILING CABINET - Look in your folders
           7   ADMINISTRATION
           8   LEAVE HPMAIL - Sign off and end the program
           9   Sign off, and Sign on as someone else

Please type a number to indicate your choice.
Type HELP if you need help at any time.

HPMAIL >
```

Fig. 5. *HPMAIL main menu display.*

```
IN TRAY of Ian Fuller                                   02/07/83 1349.

4 messages.

                                                        N A P U
                                                        E C R R
Item  Subject                  From              Received  W K I G

  1   Travel Plans             WILLIAMS,PETER    02/06/83
  2   Security                 DRAPER,COLIN      02/07/83  *     *
  3   Publications             MUELLER,AMY       02/07/83  *
  4   Part Numbers             LOVETT,JULIE      02/07/83  * *

IN TRAY >
```

Fig. 6. *Typical in tray display for HPMAIL.*

components, which can broadly be broken down into initialization, user sign-on, environment handlers, sign-off, and shutdown. The initialization module opens the data bases, the user's input and output files, and the message catalog and reads the dictionary from a disc file onto the user's stack. It initializes the mail common area with all relevant data and generally prepares the program for execution. It is also responsible for checking that the software has not been stolen and that incompatible versions of the software are not being used. The last task it performs is to print the HPMAIL banner on the user's terminal.

The user sign-on module is the first that actually solicits input from the user (by asking for a name). The name allows the user's user record to be located in the LOCAL-USER data set of the LOCAL data base, and from that point on, the user interface can go to a specific point in the object hierarchy shown in Fig. 4.

After a successful sign-on, control is passed to the main menu processor, the first and simplest of the environment handlers. This simply displays the list of options available to the user, along with the number of newly delivered messages in that user's in tray. The user is then prompted for a choice of area (environment) to enter next. This choice can be made by pressing the appropriate function key or by typing the corresponding number.

The other environment handlers for the in tray, out tray, pending tray, work area, list area, and filing cabinet, are very similar in structure. They consist of five basic states: initialization, building a list file as a temporary MPE file of the current contents of the area (e.g., all the messages in the in tray), producing an initial LIST of the area to give the user the information about its contents, prompting the user for a command, and executing the command.

Each command (READ, FILE, etc.) has its own executor, which handles the operation of that command in all areas. Once the environment handler has recognized that a certain command has been entered by the user, it passes control to the command executor for that command. It is the responsibility of the executor to determine whether the command is correct, possibly asking the user for additional information if necessary. If the command is correctly formed, the executor passes control to the action procedure for the command. This is often just a simple mail procedure. For example, the task of the FILE command executor is to determine what the user wishes to file, and where. This is resolved internally into a pair of internal item numbers in the ITEM-HEADER data set. These are presented to the action procedure for the FILE command, which is the mail procedure MATTACH, whose task it is to perform the required attachment, linking, say, an in tray message to the incoming day folder in the user's filing cabinet.

Command-Based User Interface

We are frequently asked why we adopted a command-based user interface for HPMAIL when many of Hewlett-Packard's Interactive Office products have standardized upon point-and-push interfaces, that is, the user positions the display cursor and pushes a function key. This question caused a considerable amount of discussion in the development and marketing teams during the design phase of the product and we finally decided in favor of an unsophisticated command-based interface for two main reasons.

First, we realized that many of the potential users of our product would be managers, working away from their desks, perhaps in hotel rooms, on hard-copy terminals connected to their office HP 3000 by a slow-speed telephone line. We thought that these users would derive a major benefit from using HPMAIL and so we designed our product to be no more demanding of a terminal than that it be teleprinter-compatible. We would not go so far as to claim that the HPMAIL user interface works well on every terminal that can be connected to an HP 3000, for we have not tried them all. However, we would be very interested to know of any upon which it does not work.

The other main reason for adopting the command-based interface is that, for regular users, such an interface is far quicker to use than one where you have to cycle through several screens just to access the menu you require. Users often comment that, as long as you are not making errors, the user interface is extremely terse. It does not usually ask for confirmation of actions, nor indulge in long dialog about exactly what to do. This agrees with our belief that an electronic mail system is of little use to an occasional user and to derive major benefit from it, it must be used regularly. Otherwise users would cease to rely on it as a timely communication medium. Regular users, we believe, do not want verbose or long-winded user interfaces, they know what they want to do and do not want the software to get in the way.

Ideally, of course, we would have written a user interface for every level of user who might encounter our product. However, we compromised with one we thought would

```
IN TRAY > READ 2

Start of Item 2.

Message.                                    Dated: 02/07/83 at 1135.

Subject:  Security

Sender:  Colin DRAPER / HP1600/01                        Contents:  2.

Part 1.

 TO: Ian FULLER / HP1600/01

Part 2.

Could you arrange for me to have a key to the building, please.
Regards, Colin.

End of Item 2.

IN TRAY >
```

Fig. 7. *Reading the second message of the in tray list shown in Fig. 6 gives this display.*

please the maximum number of people most of the time.

One drawback of a command-based user interface is that it is often perceived as being harder to learn initially than point-and-push interfaces. Many of our users are busy managers who have no time to go on long product training courses or to wade through long manuals just to learn how to send a simple message. I was told once while I was testing the product with some Hewlett-Packard managers in the United States that if they could not learn the essentials of the product in a maximum of thirty minutes it would not be accepted. To this end we developed an on-line training package for the product, known as the interactive training facility. This gives users the opportunity to learn about those aspects of the product they need to use at their own speed in their own time. We find this to be an extremely valuable facility in getting the product used throughout the company.

It is interesting to note while discussing user interfaces that the HPMAIL configurator program has a point-and-push user interface based entirely on VPLUS/3000. We felt that the configurator was an ideal application for such an interface and makes the potentially complex process of data base building, configuration, modification, and expansion far easier to achieve for system administrators.

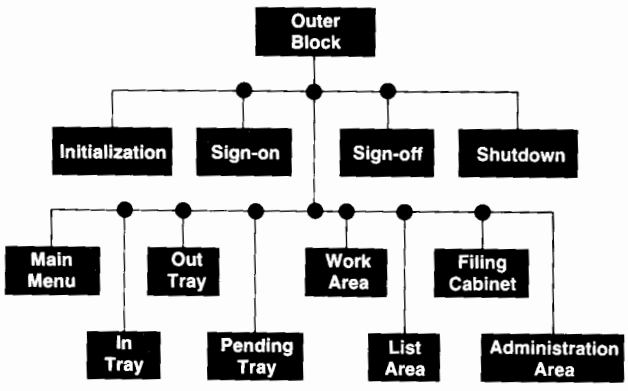


Fig. 8. *The HPMAIL user interface program has a tree structure that closely follows the object hierarchy.*

Transport System Overview

As stated previously, one of the major product objectives for HPMAIL was to provide a system in which the end user did not need to become involved with the intricacies of message routing and transmission, understanding DS/3000, or dealing with line failures, message congestion or any of the many other problems that can befall a multicomputer messaging system.

Once a user MAILs a message it is removed from the out tray by the mailroom and handed over to the transport system for sorting, transmission, and delivery. The user can track the progress of the message by setting an acknowledgement on the message and monitoring it in the pending tray while it is sent to all the recipients, wherever they may be.

Fig. 9 shows an overview of the data and control flow required to implement the store-and-forward system used in HPMAIL.

The transport system is implemented by background jobs communicating via IPC files. The jobs are known as the mailroom, transport manager, and at least one master truck communicating with a slave truck on a remote computer.

The mailroom has two basic functions, first to collect messages from users' out trays, and second to deliver incoming messages to recipients on message distribution lists where the users are supported on the same computer. It also handles delivery acknowledgments, autoforward, and autoanswer. The mailroom is structured internally in two halves. The outgoing half deals with mail collection and sorting, and the incoming half deals with local mail delivery.

Outgoing mail originates from the user interface in response to the user's MAILing a message created by the SEND, FORWARD or REPLY commands. The IPC message sent to the mailroom when a message is MAILed identifies the specific message in an out tray that has been committed for trans-

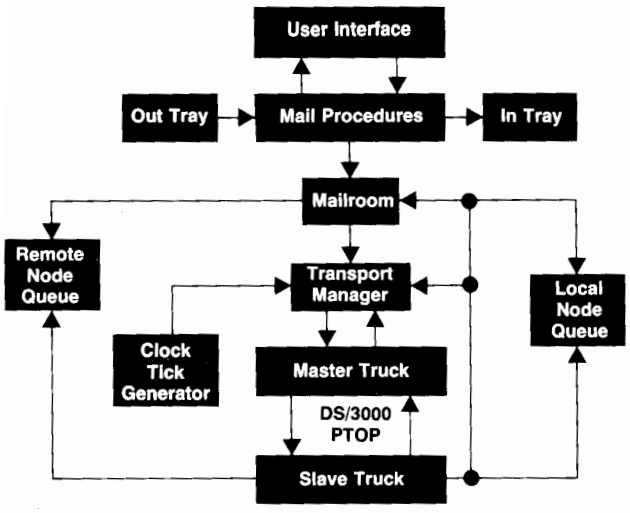


Fig. 9. *Overview of the data and control flow required to implement the store-and-forward system used in HPMAIL.*

mission. The mailroom responds by accessing the message and scanning its distribution list. It builds an internal list of the number of unique mail nodes (local or remote) that the message is bound for and then attaches the messages to the node queues for the correct destination nodes. For locally bound messages the outgoing mailroom sends an IPC message to its incoming counterpart to deal with the delivery. For remote messages it sends a message to the transport manager alerting it of the new mail. If an outgoing message requires any acknowledgment, the mailroom attaches the message to the sending user's pending tray, ready to receive the acknowledgments as they come in. The last action it performs is to detach the message from the sending user's out tray.

The incoming half of the mailroom deals with mail originating from two sources. First are the local messages sorted by the outgoing half. Second, the incoming mailroom receives IPC messages from a slave truck operating on its computer to handle messages received from a remote computer and to be delivered locally.

In each case, the incoming mailroom examines the message distribution list to determine to whom the message is to be delivered. It then locates the required users' in trays on the LOCAL data base and attaches the message to them. It also has to handle the various types of acknowledgment that can be received, amending the user's pending tray copy of the message with the fact that the acknowledgment has been received. Its other main tasks are to generate delivery acknowledgments for the message if necessary and to perform any autoforward or autoreply that a receiver has set.

Transport Manager

The transport manager is another background job. Its function is to control and schedule the operation of HPMAIL in sending mail to remote computers. It maintains tables of mail nodes for which mail is waiting and the possible routes that it can use to move the mail. This information is derived from the HPMAIL system configuration specified using the configurator program. Every fifteen minutes the transport manager receives a clock tick from its son process, the clock tick generator. This causes it to reexamine its priorities and reschedule the master trucks under its control if necessary.

The transport manager is constantly striving to ensure that the remote mail nodes with the highest-priority mail to send are being serviced by the available master trucks. Once it selects an available mail node, together with an available route to another computer, it sends an IPC message to a free master truck instructing it to establish a DS/3000 connection with another HPMAIL computer. Once the master truck has reported that it has done this successfully, the transport manager instructs the truck to clear a certain remote node queue. Once this has been done, the master truck reports back, ready to receive further instructions.

The transport manager is capable of controlling up to eight master trucks simultaneously, although it would be a very ambitious network that required this level of resources.

The master truck receives IPC instructions from the transport manager and handles all the interaction with DS/3000 and PTOP to move mail physically between computers.

The first instruction received by an idle master truck is to establish connection with a certain remote computer (identified by name) using a specified DS/3000 line, and if necessary, a specified telephone number or public data network node name. It uses an MPE DSLINE command to open a line to the remote computer, followed by an MPE REMOTE HELLO command to establish a remote session. It then attempts to create and activate a slave truck on the remote computer, using the DS/3000 intrinsic POPEN. If the slave truck is successfully awakened the master truck determines that it is, in fact, communicating with the correct remote computer by interrogating the slave for its computer name. Once this protocol has been observed, the master truck reports back to the transport manager "truck idle with DS line open" and awaits further instructions.

The next command received by the master truck in the message transmission operation is the move mail command. This instructs the master truck to transmit mail from a given remote node queue on the LOCAL data base. The master truck is responsible for serializing messages from the data base and packing them into transmission buffers for sending to the remote computer via the PWRITE PTOP intrinsic.

At the other end of the communication link the slave truck accepts these transmissions, if possible, and recreates the message on its computer. After the message has been completely transmitted, the slave truck deserializes the message onto its data base. Depending on the ultimate destination of the newly received message, the slave truck either alerts the mailroom for locally bound messages or its own transport manager for messages that are destined for onward transmission. Once the master truck has sent a complete message successfully, it detaches that message from the remote node queue, since responsibility for it has now passed to the next computer in the chain.

This process is repeated for all the messages that the master truck has to send. Once the master truck has complied with its instructions to move mail, it reports back to the transport manager "truck idle with DS line open." The

transport manager either responds with an instruction to begin transmitting messages bound for another mail node, taking advantage of the open DS line, or instructs the master truck to terminate the link if there is no more mail to be sent on that route.

The main challenge in designing the transport mechanism was to provide for all the failures that can take place during transmissions over communications lines. Examples of the type of problems that have to be handled are:

- Inadvertent connection to the wrong computer, perhaps because of an incorrectly dialed telephone number
- Finding that the remote data base cannot accommodate the message to be sent, either temporarily because of congestion, or semipermanently because the data bases are not sufficiently large
- Configuration errors resulting in transmissions being refused or an infinite loop being set up
- Line failure during any stage of the process.

Since detachment from a node queue is the last operation that the master truck performs, and this happens only after it has been determined that the message was transmitted successfully, it is unlikely that messages will be lost because of line failure. However, it is possible that HPMAIL will send messages twice in some circumstances.

Why Use PTOP?

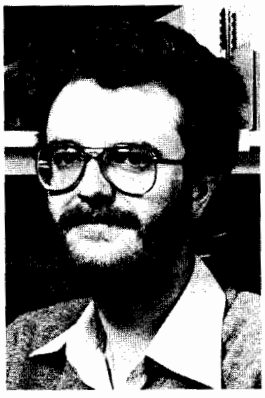
We chose the program-to-program intrinsics provided by DS/3000 in preference to the other methods of transmitting files across DS (distributed systems) connections for four reasons. First, PTOP ensures that there is a communicating process at the other end of the DS line to receive the mail. Without it, we would have had to implement yet another level of IPC communication to ensure that the remote computer was aware of the mail being delivered to it. Second, the performance of PTOP is superior to, for example, remote data base access (RDBA), since in good circumstances, the master truck can be assembling transmission buffers for the next transmission at the same time DS is taking care of the transmission of the previous buffer over the line. This makes optimum use of the communications line.

Third, PTOP provides very good application program control over the communication activity with the provision of a tag field in the PTOP intrinsics that can be used to send status, control, and block-level acknowledgments at the same time as the buffer transmissions. Lastly, the PTOP intrinsics are implemented on other Hewlett-Packard computers, for example the HP 1000 Technical Computers, giving us the option of using these machines as mail nodes in a mixed DS network.

By removing the transport mechanism to a completely separate layer of the product we will be able to take advantage of enhancements to Hewlett-Packard's data communications products as they become available with minimal disruption to the product's users. It is notable that the HPMAIL transport mechanism uses only supported features of DS/3000 and works entirely in user mode. This, we feel, contributes greatly to the robustness of the product. This remark applies to the product as a whole. In fact the only line of privileged code in the product is one where the system cold load identity is obtained. This is required to maintain system integrity, since we have to know if there has been a system restart since the data bases were last accessed.

Acknowledgments

The HPMAIL project was written at HP's Commercial Systems Division in Pinewood, England. I would like to acknowledge the contributions made by the development and marketing teams, particularly Peter Williams, who was the project manager for the first release of the project and responsible for most of the basic design. Trevor Wing, our product manager, was in charge of the marketing of HPMAIL and a constant source of encouragement throughout its development. I would also like to acknowledge the help given by users, both in England and the United States, who agreed to test the product and provide us with their reactions. Particularly helpful in this area was the office utilities group at HP's corporate headquarters in Palo Alto, California, who willingly gave us the benefit of their long experience in implementing computerized office systems within Hewlett-Packard.



Ian J. Fuller

Ian Fuller received the BSc degree in mathematics, computer studies, and physical sciences from Oxford Polytechnic, Oxford, England in 1978. After two years working on software for message switching and PBX telephone systems, he joined HP in 1980. At HP, he has been involved with the HPMAIL project as a development engineer and a project leader and currently is the HPMAIL project manager. Born in Gosport, Hampshire, Ian now lives in Little Sandhurst, Camberley, Surrey, four miles from his office at HP's Commercial Systems Pinewood software development center. His interests include photography and travel.

Integrated Tools Improve Programmer Productivity

This software subsystem for the HP 3000 Computer System saves program development time by giving the programmer access to several utilities through a single command interpreter.

by Anil K. Shenoy and Carolyn M. Bircher

HPTOOLSET IS A SOFTWARE SUBSYSTEM that provides an integrated program development environment to improve programmer productivity. Through a single, friendly user interface, it offers programmers access to several programming tools needed for software development. It lets programmers perform multiple functions simultaneously and access information about other program development functions being performed.

HPToolset runs under the MPE operating system on the HP 3000 Computer System. The first release is geared towards improving COBOL II programmer productivity but HPToolset is structured such that many of the features do not depend on any specific language.

The phases of implementing a software program include source code creation, modification, compilation, execution, and testing. These steps are executed repeatedly during the life of a program and can be very time consuming. HPToolset relieves the programmer of the responsibility of manually directing program flow through each of these steps and minimizes development time by allowing the programmer to move between the different phases of program development without leaving the HPToolset environment. Since program development is often a team effort, HPToolset promotes sharing of information between programmers, which in turn increases the effectiveness of the programming effort.

HPToolset offers the following aids towards improving productivity:[1]

- Menu and labeled function key interface
- Ability to perform multiple functions without exiting and reentering subsystems
- Sharing of files between team members through version management
- Full-screen editing with a point-and-push philosophy (i.e., the user positions the cursor and pushes a function key)
- Background compilations and on-line listings
- Point-and-push method to find compilation errors
- A GO key to eliminate the manual steps between compilation and execution
- Dynamic symbolic debugging
- Ability to go directly from symbolic debugging to the related compilation or source listing
- Context-sensitive HELP facility.

Basic Concepts

The following concepts are fundamental to an understanding of HPToolset.

A workspace is a directory of the files that are used in the development of a program. It represents the programmer's development environment. The programmer is required to create or reference a previously created workspace before executing most HPToolset functions.

The HPToolset access method, TSAM, is a customized file access mechanism that is built on top of MPE's file system. It provides, among other features, the ability to define and access versions of changes made to a source file. HPToolset provides interface procedures for compilers and other procedures that wish to access TSAM files. The COBOL II compiler uses these procedures to read TSAM source files. A conversion utility is also provided to convert MPE ASCII files or KSAM/3000 (keyed sequence access method) files to TSAM format. Only TSAM files can be edited via the HPToolset editor.

When a source file is compiled it is translated from a higher-level language, like COBOL II, to a lower-level language of machine instructions. Source files are compiled into a user subprogram library (USL) file. MPE's segmenter is called by HPToolset when the USL file is to be prepared into a program file. The segmenter binds all the needed routines to the program so they can be found by the system when the program is executed. Each workspace has

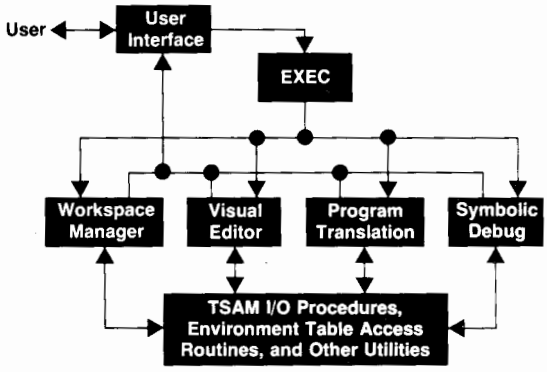


Fig. 1. *HPToolset program structure. All activities are managed by the EXEC. TSAM is the HPToolset file access method, which allows multiple versions of a file to reside within the same file.*

one USL and one program file associated with it. Multiple source files can be compiled into a single USL file.

Message files are used for communication between processes. They act as a first-in-first-out queue of records with one process writing and the other reading. If the queue is empty, the reader process is put into a wait state until a new record is written. When a record becomes available, the read is satisfied and the record is deleted from the queue.

The HPToolset EXEC

Fig. 1 shows the basic components of HPToolset. All activities that occur within HPToolset are managed by the EXEC. It interprets the user input, determines if it is valid for the current context, and calls the appropriate processor to act upon it. By having all of the HPToolset utilities operate under the umbrella of a single command interpreter, a consistency is achieved which has not existed between the unrelated tools previously provided on the HP 3000. The user need learn only one interface and time is saved by not having to exit from one utility before entering another.

The EXEC also makes it possible for the programming tools to work together. It allows several functions to be active at one time. For instance, a compiler listing can be displayed at a debug breakpoint, or one file can be read while another is being edited. This relationship between the tools is maintained by means of an environment table. This table includes information about all current HPToolset activities. It is used by the EXEC to ensure that commands are not executed in an unsuitable environment. It also makes it possible for the user to see a list of all activities related to the current workspace by issuing a single command.

Because the EXEC can determine what other functions are currently being performed, it can also provide context-sensitive defaults. For instance, if the Edit function key is pressed while a program is being debugged, the user will not be prompted for an edit file name. Instead, the source file related to the program being debugged will automatically be displayed such that the first source line on the screen corresponds to the current breakpoint location. Also, when the HELP key is pressed, it will display the HELP screen most appropriate for the current context. If the error has just occurred, it will give further explanation of the message and/or a corrective action. Otherwise, an overview of the most recent function is provided.

Even though the environment table resides in the global variable area, it is not accessed directly by any of the HPToolset command processors. Special-purpose routines were written to retrieve and update information in the table. By requiring all processors to access the table through these routines, control is centralized and a mechanism is provided to ease future debugging of the product.

TSAM

The HPToolset access method, TSAM, provides a mechanism for multiple versions of a file to reside within the same file instead of having to define a different file for each set of changes.

During the program development cycle each source file goes through many changes. These changes are typically cycled through compile, prepare, and execution iteratively until the programmer is satisfied that the changes are stable. At this point the programmer may wish to freeze the source so that others can share it. Before HPToolset, to make further code modifications the programmer would have to copy the contents of the frozen source to another file so modifications could be made to it while the team members accessed the frozen source file. With HPToolset, this function can be done by establishing a new version of the file by an HPToolset command and designating the frozen source as the reference version, that is, the version that others will access by default.

For example, let us assume that a file goes through the following changes:

Version 1	Version 2	Version 3
a	a′	a′
b	-	b′
c	c	c
-	d	d′

Here a, b, c, and d represent COBOL II source lines. Line a was changed in version 2 but not in version 3. Line b was

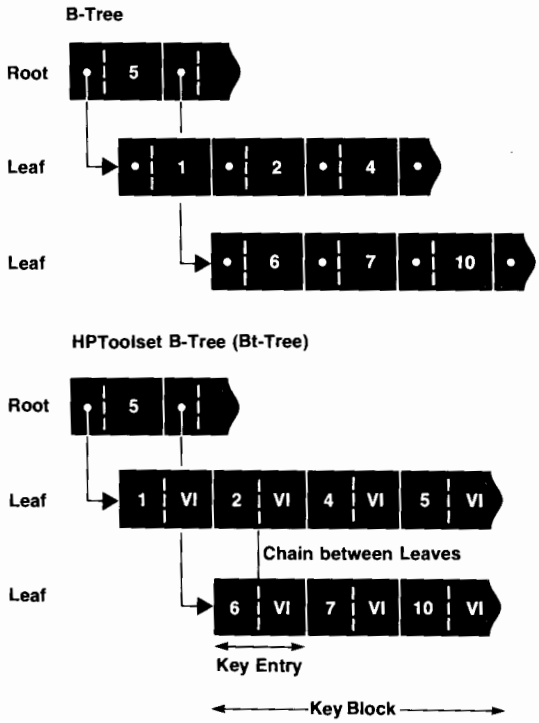


Fig. 2. *HPToolset supports both random (keyed) and sequential file access. 12-byte alphanumeric keys are organized in a modified B-tree structure called a Bt-tree. The trees shown here have only two levels, but in general there are many intermediate levels between the root and leaf levels. At each level are one or more key blocks containing several key entries that include pointers to lower levels. Blocks at higher levels are considered parent blocks of the lower-level blocks they point to. In the HPToolset Bt-tree all keys are represented at the leaf level to aid sequential access, and each leaf key entry contains variable-length version information (VI), including pointers to the data associated with the key value.*

deleted in version 2 and added again in version 3. Line c was left unchanged. Line d was first introduced in version 2 and modified in version 3.

Before HPToolset, these versions had to be saved in three separate files. HPToolset saves the changes associated with each version of the file and at retrieval time returns all the records relevant to the version being accessed. Logically, therefore, the user sees each version as a separate file, but physically they are all in the same file. This saves disc space and eliminates the bookkeeping of associating changes with file names.

When a TSAM file is created it has one version—version 1. This is also the latest version. After the development cycle referred to earlier, the programmer freezes version 1. The latest version is automatically updated to version 2. Only the latest version, version 2 in this case, can be modified. The programmer can set version 1 as the reference version so that other users will access version 1 by default. After making further changes to version 2, the programmer may wish to freeze it. Version 3 is now the latest and is the only one that can be modified. The programmer may choose to have version 1 continue to be the reference version or move the reference to version 2. While the latest version is being modified other versions of the file may be read concurrently. HPToolset also provides commands to purge one or more versions of a file, to label the versions with comment strings, and to list the changes between versions.

TSAM supports both random (keyed) and sequential access. Data records are stripped of trailing blanks before storage. The keys are 12-byte ASCII alphanumeric keys and are organized in a modified B-tree structure.[2] For purposes of this discussion we will refer to this modified structure as a Bt-tree. The B-tree structure was modified primarily to:

- Optimize sequential access to the file without sacrificing random access performance. This is because the majority of the accesses made by the HPToolset editor are sequential. This is especially true in visual edit mode where blocks of data are read for screen display. The Bt-tree structure attempts to minimize tree traversals, thereby minimizing key block swapping. This is important when constraints do not allow multiple key blocks to be memory-resident concurrently.
- Keep version-related information with the key indexes.

Fig. 2 shows the B-tree modifications. In Fig. 2, the numbers symbolize key values and VI represents variable-length version information, including pointers to the data associated with the key values. The major differences between the B-tree and Bt-tree structures are:

- Each leaf entry in the Bt structure consists of the key and version information, which is of variable length. In the B-tree each key entry has a fixed length.
- Leaf key blocks are chained forward and backward in the Bt-tree.
- In a B-tree both leaf and nonleaf entries contain pointers to the next lower level. Even though these are unused in the leaf (since the leaf is the lowest level), space is reserved for them. This is avoided in the Bt-tree, saving wasted space.
- In the Bt-tree all keys are represented at the leaf level. This is to provide for sequential traversal without having to read in the parent blocks. Although this introduces data redundancy, it improves sequential access time, especially if multiple key blocks cannot be held concurrently in memory.

Workspace

A workspace is a TSAM file with one version. Each data record is 80 bytes long. The workspace is a directory of the files that pertain to the programmer's development environment.

A file is considered owned by a workspace if it is created via the HPToolset editor, converted from ASCII or KSAM, or copied from another workspace while the programmer is associated with this workspace.

A file is considered used if the programmer wishes to share it without making a copy of it. In using a file the programmer will access the reference version of that file by default, but can explicity override the default and access other versions.

A workspace contains information about owned and used files, such as the total number of active versions in the file, the reference version of the file, the source language (currently COBOL II or "other"), and whether it is a main program or a subprogram. In addition, the names of the USL file, the program file, and on-line listing files are maintained in the workspace. The programmer can display this information via the Show Files and Program commands. Fig. 3 and 6 show these displays.

Information in the workspace is partitioned by key ranges for easy access. Workspace control information is in a range

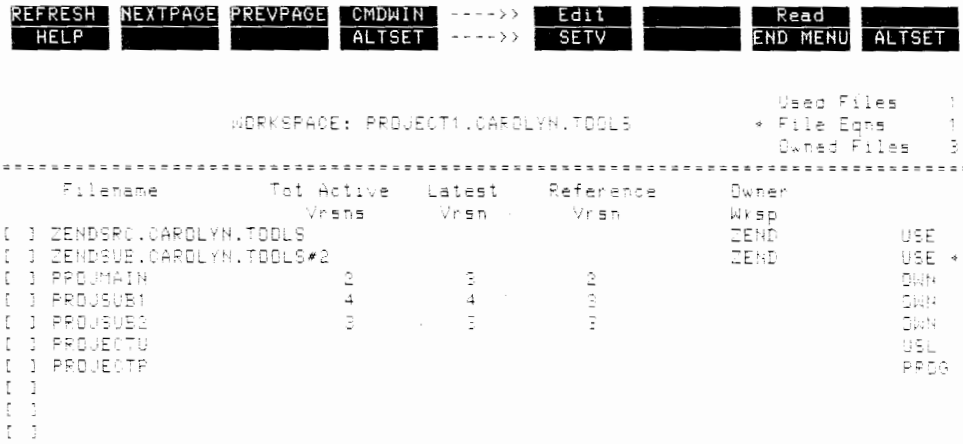


Fig. 3. *The Show Files command displays information about the files owned and used by a workspace.*

from 000001000000 to 000002000000. Used file information lies between 000002000000 and 000003000000, owned file information lies between 000003000000 and 000004000000 and so on.

When HPToolset executes a command that opens a source file, it reads the workspace to determine if the file is used or owned. If it is owned, the latest version of the file is accessed by default. If it is used, the reference version is accessed by default. During compilation and execution, the workspace is read to obtain the names of the USL and program files.

User Interface

The HPToolset user interface is designed to be friendly and yet provide the flexibility required to support multiple utilities. An effort was made to make the commands natural and to prompt the user for omitted parameters rather than giving a syntax error and requiring that the entire command be retyped.

The terminal's function keys (also called softkeys) are programmed to correspond with most of the commands. The function keys are designed so that there are always two sets available, with eight keys in each set. One set, known as the permanent set, always remains the same. It contains functions that are relevant in most HPToolset environments. The key functions in the other set change as the environment changes so that the most useful set is present. One key in every set is reserved for switching back and forth between the permanent and context-sensitive softkey sets.

The function key labels for at least one set are always displayed on the terminal screen. On HP 262x-series terminals, the labels for the active set are displayed in the built-in template at the bottom of the screen. Both the permanent and context-sensitive softkey sets are displayed side by side at the top of the screen on HP 264x terminals, as shown in Fig. 4. Arrows between the sets indicate which is active. The labels are displayed in a highlighted template, which is protected by memory lock. To change the labels, memory lock is turned off, the cursor is moved to the top of the screen to write the new labels, memory lock is turned back on, and the cursor is returned to its original position so that no other contents of the screen are changed.

VPLUS/3000 was used to implement the menus for HPToolset. Instead of having the user press the **ENTER** key to transmit input, the VPLUS/3000 autoread option is used. Appropriately labeled function keys are provided with all menus, and when one of them is pressed, HPToolset does the autoread to pick up the terminal input before performing the desired function. This provides a menu interface consistent with the user interface for the rest of the product.

Editor

HPToolset includes a visual editor. It displays a portion of the edit file on the screen. This can be edited directly with the terminal's cursor movement and editing keys. The terminal is strapped so that all cursor actions are transmitted to HPToolset. The movements are recorded into a buffer which is evaluated when the return key or any function key is pressed, or when it is full. All changes recorded in the buffer are written to the edit file with each buffer evaluation to ensure data integrity if a system crash should occur during editing. An UNDO command is provided to allow the user to cancel the last changes written to the file.

Function keys are also used in the visual editor to implement movement from one location to another in the edit file and to facilitate editing functions, such as MOVE and COPY, that cannot be done with the terminal editing keys. One of the keys, labeled MARK, is used to select lines to be moved or copied and to select the destination. Once the lines have been selected, the operation can be completed by simply pressing the MOVE or COPY function key.

Since a language is associated with all TSAM source files, the editor is able to set editing options to the values most appropriate for each file. The HPToolset editor sets tabs and margins differently, depending on the language associated with the file. Of course, any of the options set by HPToolset can be overridden by the user.

Fig. 5 shows a typical visual edit screen.

Program Translation and Execution

As mentioned earlier, after a source file is created, a number of translation steps are needed before it reaches a stable state. The steps are compiling, preparing, executing, and debugging.

When a workspace is entered, HPToolset provides default names for USL and program files. The programmer can specify different names and the options to be used while preparing the USL file. By issuing the Program command or pressing the function key, the programmer can display compile-related information on the source files in the workspace. This is shown in Fig. 6. The programmer can now mark the file(s) to be compiled and press the Compile function key. HPToolset maps this into one or more calls to the COBOL II compiler.

The compile process is created in background mode so that HPToolset is active while the compiler is executing. HPToolset sets up file equations with the formal names that COBOL II expects for the source, USL, listing, and workspace files. Three message files are used to communicate information during compilation: let's call them MF1, MF2, and MF3. The compiler directs its listing to MF1 and completion statistics such as number of errors and compilation time to MF2.

If the file being compiled is owned by the workspace, an on-line listing is generated. If not, the listing is sent to the printer. HPToolset reads MF1 and transfers data to a TSAM

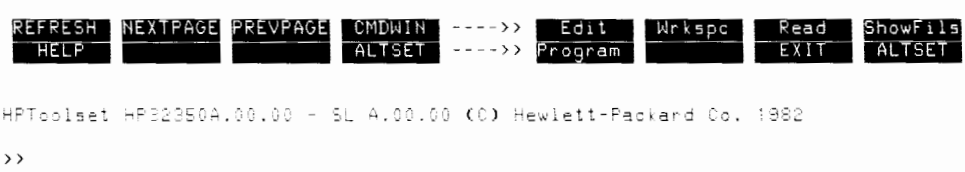


Fig. 4. *HPToolset provides two eight-key groups of function keys (softkeys), a permanent set (left) and a variable set that changes with the environment. On terminals capable of displaying both sets, arrows indicate which set is active.*

```
REFRESH  NEXTPAGE  PREVPAGE  CMDWIN  ---->>  SET EDIT          Prog Mod  Listing
 HELP                       ALTSET  ---->>  Browse   PRINT    END EDIT  ALTSET

  1     $CONTROL SUBPROGRAM, SYMDEBUG
  2      IDENTIFICATION DIVISION.
  3      PROGRAM-ID.
  4          COMPUTE-SUB.
  5      AUTHOR.
  6          JOE PROGRAMMER.
  7      DATE-WRITTEN.
  8          5-27-81.
  8.01
  8.1    ENVIRONMENT DIVISION.
  8.2
  9      DATA DIVISION.
 10      LINKAGE SECTION.
 12      77  WEEKLY-SALARY       PIC  999.
 13      77  AMOUNT-OF-SALES     PIC  9999.
 14      77  TOTAL-EARNINGS      PIC  9999V99.
 14.1
 16      PROCEDURE DIVISION
 17          USING WEEKLY-SALARY AMOUNT-OF-SALES TOTAL-EARNINGS.
 23.2    SUB-PARAGRAPH.
```

Fig. 5. *A typical HPToolset visual edit screen.*

file until there is no data left to read and there is no writer associated with MF1. This signals completion of the compiler's listing activity. The programmer can scroll through an on-line listing interactively, mark a line or an error if any, and have HPToolset display the corresponding line in the source file. Pressing the PRINT key produces a hard copy of the listing.

Completion of compilation is signaled by MF2's having data in it and no write activity. HPToolset informs the programmer that the compilation is completed, via the user interface routines. It now checks to see if another file is to be compiled (i.e., more than one file was marked when the user pressed Compile). If so, information on this compile is passed via MF3. Compiler creation and the subsequent sequence of events is repeated.

After compilation the next step is to prepare the USL file into a program file. This is done by pressing the PREP key. Any prepare options specified in the workspace are now honored. Alternatively the user can type in the PREP command and specify the options. HPToolset now creates a temporary file and writes the MPE segmenter commands necessary to perform the PREP function to it. It then invokes the segmenter and directs it to get its input from the file.

The final step is to execute the program file. This is done by pressing the RUN key. The program executes in foreground mode. If symbolic debug was not requested, HFToolset is inactive until program execution is completed. If symbolic debug was requested, the program pauses at the first executable statement and control is returned to the programmer so that execution can be monitored.

The programmer can combine the compile, prepare, and execute steps by pressing the GO function key. To do this the programmer first issues the Program command to display the source files in the current workspace, then marks the files and presses the GO key. The file(s) are compiled, prepared into the workspace-defined program file, and executed.

Symbolic Debug

The debugger in HPToolset allows programmers to set breakpoints, trace program flow, display variable values, and change the values of variables by using the symbolic names used in the program source. Debugging time is saved by eliminating the need to determine the octal addresses of program locations or data items before accessing them in the debugger.

To implement this, it was necessary to save additional information in the USL (object) and program files. The COBOL II compiler was modified to store the symbol table information into the USL file. Three new USL header types were added to hold this information, two to hold the PMAP information* and one for all other symbolic information such as the addresses associated with data items and program statement numbers.

The compiler also inserts a PCAL (procedure call) instruction at the beginning and end of the main program and each subprogram, and at the beginning of each section and paragraph. These instructions pass control to SDB'SPY,

*PMAP = program map, a list of procedures within a program.

```
REFRESH  NEXTPAGE  PREVPAGE  CMDWIN  ---->>  Edit     Compile  SET PROG  NO PRINT
 HELP                       ALTSET  ---->>  GO       Listing  END MENU  ALTSET

   SOURCE FILES for Workspace PROJECT1.CAROLYN.TOOLS

      PROGRAM ID                    LANG    TYPE    SOURCE FILE      LISTFILE

[ ] SALARY-PROG                     COBOL   MAIN    PROJMAIN         PROJ1687
[ ] COMPUTE-SUB1                    COBOL   SUBP    PROJSUB1         PROJ1612
[ ] COMPUTE-SUB2                    COBOL   SUBP    PROJSUB2         PROJ1632
[ ]
[ ]
[ ]
```

Fig. 6. *The Program command displays compile-related information about a workspace's source files. To compile a program file, the programmer simply types any character in the box next to the program ID and presses the Compile key.*

which is a procedure that acts as the communication link between the HPToolset debugger and the user program. The PCALs are used by SDB'SPY for tracing program flow.

During the linking process, the symbol table information is passed from the USL file to the program file. The segmenter creates a son process, called SEGSYM, which takes the information from the USL file, sorts and reformats it, and adds it to the program file. Several new MPE intrinsics (routines) can be called by HPToolset to access this information for converting symbolic names to the corresponding octal addresses.

Of course, it is not necessary to include this extra information in program files that are not going to be run with symbolic debug. By default it will not be added to the USL file by the compiler. A $CONTROL SYMDEBUG statement must be included at the beginning of the source files for which the symbolic debug information is desired. The segmenter will automatically pass it from the USL file to the program file unless the NOSYM parameter is included in the PREP command. Including the NOSYM parameter will also cause all of the inserted PCALs to be converted to NOP instructions.

As soon as HPToolset creates a user process to be run with symbolic debug, it does a handshake with SDB'SPY. Communication between HPToolset and SDB'SPY is executed via three message files. HPToolset sends the names of the message files in which it will write and receive messages (MF2 and MF3) to SDB'SPY in MF1. It then initiates a read from MF3 and waits for a reply. Because of the inserted PCAL instruction, SDB'SPY gets control as soon as the user program is created. SDB'SPY returns the handshake by writing a message to MF3 to inform HPToolset that it is ready to receive a debug command.

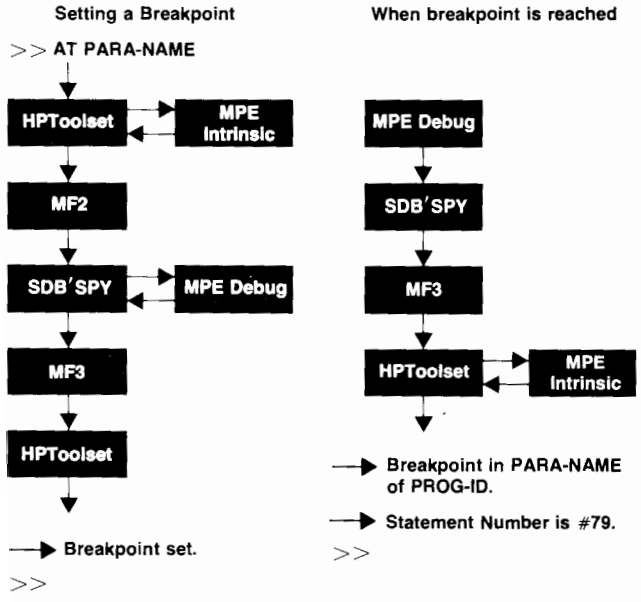


Fig. 7. *Intrinsics (routines) were added to the HP 3000 operating system, MPE, to facilitate converting between symbolic names and octal addresses. Intrinsics were also added to call the MPE debug facility programmatically for setting and clearing breakpoints. MF2 and MF3 are message files used by HPToolset.*

When the user enters a symbolic debug command to set or clear a breakpoint, HPToolset determines the PMAP name and offset for the symbolic location or statement number and passes them to one of the new MPE intrinsics. The intrinsic converts them to the corresponding octal address. HPToolset then passes the address to SDB'SPY through MF2. SDB'SPY calls the MPE debug utility programmatically to set or clear the breakpoint. This capability was added to MPE debug by the addition of two intrinsics, SET'BREAKPNT and CLEAR'BREAKPNT. When the operation has been done, SDB'SPY returns the status to HPToolset in MF3. Fig. 7 diagrams this flow of events.

When the user resumes program execution, HPToolset sends a message to SDB'SPY and then initiates a read from MF3. This puts HPToolset in a wait state until SDB'SPY satisfies the read. One of the parameters passed to the SET'BREAKPNT intrinsic is an address in SDB'SPY where control will be passed when the breakpoint is reached. When SDB'SPY gets control at a breakpoint, it writes the program location to MF3 to satisfy the read which was issued by HPToolset. HPToolset then converts the octal breakpoint location to the symbolic equivalent, displays it, and prompts the user for a command.

When a DISPLAY or MOVE command is entered to display or change the value of a symbolic data item, the symbolic name is converted to an address in the data stack and a length (in words). The request is then passed to SDB'SPY through MF2. SDB'SPY can directly access the program data stack since it is called by the user program. It returns the contents of the requested locations to HPToolset where it is formatted and displayed.

Context-Sensitive HELP

As was mentioned earlier, HPToolset has a context sensitive HELP facility. This facility provides an overview of each of the utilities, specific text about each of the commands, and further explanation of error messages. The command descriptions are contained in a TSAM file, keyed on the internal command numbers which are used by the command interpreter. By using the same command numbers as are used by the command interpreter, the HELP facility can share the procedures for converting command names to numbers, and can easily find the correct syntax for the command just entered by picking up the current command number from the command interpreter. This is used if HELP is requested immediately after receiving a syntax error.

When a nonsyntax error is displayed, the error number is saved. The HELP facility uses that number to locate the error explanation and corrective action. This information is obtained from the error catalog where it has been included as comment lines immediately after the error text. After MPE's GENMESSAGE intrinsic has been called to locate the error message in the catalog, the file pointer is left in a position that makes it easy for the HELP facility to find the error and read the comment lines following it. The result might look like:

```
>>ADD 10
***No file currently open for editing. (85)
>>HELP
```

Cause: A command was entered which is only valid while an Edit file is open, and there was not one open.

Action: Do an Edit command before reentering the command.

>>

Technical Summary

The HPToolset subsystem is written entirely in SPL, the HP 3000 systems programming language. It is currently made up of 260 source files which contain a total of 130,000 lines of code. HPToolset is contained in two program files named TOOLSET and TSETUTIL. TOOLSET is the primary program file and contains 37 segments. TSETUTIL contains one segment and is created as a separate process by TOOLSET to implement background compiles and other special functions.

In addition, two new segments were added to the system segmented library to support HPToolset. They contain the I/O procedures for accessing TSAM files and SDB'SPY.

Several existing products were enhanced to support HPToolset. The COBOL II compiler, the MPE segmenter, and the MPE program loader were all changed to handle the symbol table and PMAP information in the USL and program files. The COBOL II compiler was given the ability to read TSAM source files and workspace files. Also, intrinsics were added to MPE for programmatically calling the MPE debugger to set and clear breakpoints and to facilitate the conversion of symbolic locations to the corresponding octal addresses.

Acknowledgments

HPToolset is the result of the hard work and cooperation of a dedicated team of engineers. The EXEC and editor were designed and implemented by Linda Lawson. Pat Miyamoto developed the user interface. The program translation interfaces were developed by Lynn Smith. Barbara Packard designed the symbolic debugger which was implemented by her and Alan Padula. Important contributions were also made during the final stages of the project by Paul Chan and Greg Gloss.

References

1. HPToolset Reference Manual (Hewlett-Packard Company, 1982).
2. R. Bayer and C. McCreight, "Organization and Maintenance of Large Ordered Indexes," Acta Informatica, Springer Verlag, 1972, pp. 173-189.

Carolyn M. Bircher

Carolyn Bircher joined HP in 1979 after receiving her BS degree in computer science from California Polytechnic State University, San Luis Obispo. She's done current product engineering for EDIT/3000 and FCOPY, two HP 3000 software subsystems, and she developed and is doing current product engineering for HPToolset. Born in Fresno, California, she is married and now lives in Sunnyvale, California. Her interests include square dancing and church activities.

Anil K. Shenoy

Anil Shenoy holds an MS degree in Computer Science from the University of California at Berkeley. With HP since 1979, he helped develop HPToolset and is now a software project manager. Before coming to HP he was involved with data base software development and project management for Control Data Corporation and operating system development for Xerox Corporation. Born in Mangalore, India, Anil is married, has a daughter, lives in Los Altos, California and enjoys swimming and badminton.

HEWLETT-PACKARD JOURNAL

HEWLETT PACKARD

5953-8508