Conversational Access to a 2048-Word Machine

MARY ALLEN WILKES
Washington University, St. Louis, Missouri*

LAP6 is an on-line system running on a 2048-word LINC which provides full facilities for text editing, automatic filing and file maintenance, and program preparation and assembly. It focuses on the preparation and editing of continuously displayed 23,040-character text strings (manuscripts) which can be positioned anywhere by the user and edited by simply adding and deleting lines as though working directly on an elastic scroll. Other features are available through a uniform command set which itself can be augmented by the user.

The machine, although small, aids program design by providing display scope and premarked randomly addressable LINC tapes as standard items, in an environment similar to that of a sophisticated terminal. The tapes are logically similar to a disk. Priority was given to the design of efficient tape algorithms to minimize the limitations of the small memory. Techniques developed for handling scroll editing, filing, and the layered system structure are outlined.

LAP6 is used by about 2000 people in 11 countries. Its design was strongly influenced by performance criteria established in interviews held with the LINC users themselves during the specification period.

KEY WORDS AND PHRASES: conversational computer access, display editing, display oriented system, filing algorithms, LAP6, layering, LINC, man-machine communication, on-line editing, on-line efficiency, on-line environment, scroll editing, small machine system, tape filing, tape oriented system, text editing
CR CATEGORIES: 3.73, 4.10, 4.11, 4.19, 4.22, 4.30, 4.40, 4.41

Introduction

An operating system which runs efficiently in a "conversational mode" is a necessity for a computer which is operated solely as an on-line machine. When the computer is a LINC, with 2048 12-bit words of memory, the challenge of producing a useful system forces the careful evaluation of both techniques and attitudes in order to provide operating flexibility generally deemed possible only on much larger machines. It is the intent of this paper to describe the LINC operating system, LAP6, and hopefully thereby to provide some insight into the evaluation process influencing program design in the on-line environment.

LAP6 is used to generate and edit text strings and to manipulate file entries (save, retrieve, replace, delete, or copy). Text strings formatted as LINC source programs are converted to binary machine form, filed, loaded, and checked out within the LAP6 framework.

The challenge of producing a useable system for the LINC is nothing if not enhanced by the size of the core memory. Of the 2048 words, only half are programmable. The other major factors influencing LINC program design are the machine's well-integrated in/out elements, the particular requirements of the on-line environment, and the needs of the LINC users themselves.

The Machine

The LINC [1] is easy to use and provides an intimate, uncluttered working environment. A keyboard is the main source of digital input, and a well-designed console provides sophisticated features for on-line program check-out, reducing the need for software debugging aids. The LINC is programmed almost entirely in "machine language."

The inclusion of a display scope and of magnetic tape units in a general purpose machine specification is still rare among small computers on the market. Although on-line program design is profoundly affected by the operating characteristics of associated peripheral devices, these devices are frequently not standard from machine to machine, and almost never well integrated. The result is that system programs organized around a "minimum" set of peripherals are considerably weakened in an effort to provide software which "everyone can use."

By contrast, the fact that every LINC user has at least a scope, two magnetic tape units, and a keyboard greatly simplifies a number of design choices without sacrificing sophistication to generality. The LINC tapes themselves are unusual[1] in that their random addressability makes it possible to use them the way a disk or even a larger core memory might be used in another environment; their design, which evolved from that developed for the TX-2 computer at Lincoln Laboratory [2], is remarkably well suited to symbol manipulation tasks.

The pocket-sized tapes belong exclusively to their individual users who are responsible for allocating tape contents. Operating system filing algorithms can be kept relatively simple since questions of shared files do not arise. Similarly, since the system itself resides only on its user's tape, it becomes "his" system, retaining from day to day his data and programs without resetting.

The fact that in the LINC environment there are at least as many tape copies of a program as there are users of that program has other implications affecting reliability and standardization. A program once issued is not changed

* Computer Systems Laboratory. This work was supported by the Division of Research Facilities and Resources of the National Institutes of Health under grants FR-00218 and RR-00396.

[1] The tapes are premarked under program control on the computer. Each contains 512 consecutively-numbered 256-word blocks, in fixed positions on the tape, randomly addressable by block number. Transfers between tape and memory are always handled in block units. A single LINC instruction can select a tape unit, start the tape motion, find the requested block, and make and check a transfer of from one to eight sequential blocks (one full core memory) of information. The LINC searches for a block as the tape is moving either forward or backward; it transfers information in the forward direction only.

lightly, and it must be documented well enough that its user can determine at any time whether his copy is an accurate reflection of a specified standard.

The size of the core memory has its most direct influence on system structure. LAP6 is segmented into 11 "layers" which are held on the tape and read into the memory under program control as needed. This is a fairly simple procedure with a premarked tape; if some care is taken with the segmenting and tape layout, the result need not be inefficient for the on-line user.

The small memory has had surprisingly little effect on the functional specification of the system. It was possible to include most features felt to be essential in any good machine language operating system. The necessity, however, of producing highly efficient code for the small memory accounted for a nonnegligible one third of the programming time.

The On-line Environment

The LINC environment is similar to that of on-line terminals equipped with scope and keyboard, and the LINC experience suggests what can be expected of this environment, especially when small tapes, such as the recently popular cassettes, and perhaps some memory are added to such terminals. Perhaps the most important demand made on the on-line environment is efficiency, not only machine efficiency, but a parallel component which might be called "user efficiency."

Machine efficiency is frequently regarded as a measure of response time, or the time elapsing after the user initiates one action and before he can initiate another. In the LINC environment the response time is determined not by the rate of service to one of perhaps several terminals, but almost entirely by local tape motion characteristics. Since these are directly measurable, response time can be optimized, and tape algorithms correctly dominate most LINC program design.

Solutions to questions of user efficiency probably defy optimization. However, if "man-machine interaction" for its own sake is not the goal, some effort must be expended to make proper use of the user's energies. A survey of LINC users revealed that their primary concerns, in addition to machine efficiency, were reliability and simplicity. One concise, if frustrated, user simply said, "I don't care where it files the program, I just don't want it to ask a lot of questions." A degree of arbitrariness on the part of the operating system is perhaps invited. The specific user efficiency considerations most profoundly influencing LAP6 relate to what the user can reasonably be expected to remember, what actions he can reasonably be expected to take to accomplish a task, and what control he feels he has over the program and the machine.

It is especially tiresome in the on-line situation to have to refer to documentation to recall an operating procedure detail or the significance of an "error message." Although a minimum set of functions must be provided, the threshold of both the user's memory and his patience is quickly reached. Several guidelines operated on the design of LAP6 toward the ultimate goal of providing brief documentation which can be set aside after the introductory sessions with the system [3]. The 17 system commands (Table I) have nearly identical formats with arguments of only three types, always supplied in the same order. Arguments are simply omitted if not needed. For 14 of the commands this single statement fully specifies the requested function.

A command repertory can be trimmed by including only those functions over which the user needs explicit control. Operations which properly belong to "system maintenance" are those most irritating to regular users and most unsettling to users with no prior computer orientation. With LAP6 we found that "packing" operations, text line number sequencing, tape positioning, and creating

Table I. LAP6 Meta Command Summary
(User supplies the underlined information; parentheses identify optional arguments.)

→*LN*	Move scroll to line *LN*
→Save Manuscript (*LN*, (*LN*,)) *NAME*, *UNIT*	Scroll manuscript to file; with replace option
→Add Manuscript *NAME*, *UNIT*	Filed manuscript to scroll, "inserted" at current line
→Display IndeX *UNIT*	File Index to scope; with delete option
→Copy Manuscript *NAME*, *UNIT*	Manuscript from file to file; with replace option
→Copy Binary *NAME*, *UNIT*	Binary program from file to file; with replace option
→Copy File *UNIT*	All non-duplicate file entries from file to file
→ConVert	Current scroll manuscript to current binary; with displays
→Display Symbols	Symbol table of current binary to scope
→Save Binary *NAME*, *UNIT*	Current binary program to file; with replace option
→EXit	Current status of LAP6 to tape for later access
→LOad Binary (*NAME*, *UNIT*)	Exit; binary program to memory
→Free	Exit; user's meta command layer to memory
→CoPy	User specifies tape transfers in response to displayed questions
→Print IndeX *UNIT*	File Index to teletype
→Print Manuscript (*LN*, (*LN*,)) (*NAME*, *UNIT*)	Manuscript to teletype; formatted and paged as text
→LIst (*LN*, (*LN*,)) (*NAME*, *UNIT*)	Symbol table and program listing to teletype

and erasing file indices could all be handled automatically, and usually more efficiently, by the program.

Text editing operations, probably the most important in terms of the user's time, responded dramatically to a simplification effort, the specific actions required of the user being reduced to finding his place in the text using the scope and to adding or deleting information at that place. Explicit editing commands do not appear; the user edits a text string directly without specifying the editorial function or its bounds.

A simple "NO" appears on the scope during filing operations when a file or its index is full or when a requested file entry is not found. This is the only "error" indication which LAP6 does not attempt to handle automatically and has no effect on the status of the system or the user's ability to continue. Other ambiguous situations are resolved automatically by LAP6 as defined in the users' documentation.

The sense of control a user feels he has over the system is a major factor in making him comfortable using it and can be completely destroyed by an unreliable program. Providing real reliability can add considerably to the programming investment since it involves not only removing the errors but incorporating certain generally unseen safeguards which protect the interaction between user and machine, regardless of what demands the user may make. Such things as alarming or ambiguous, even if harmless, system behavior must be eliminated; simply put, it should not be possible for the program to go into a state which the user does not understand.

The major LAP6 characteristics contributing to the user's sense of control are its documentation, its mechanical operation, and its structure. An effort was made to match closely the program's behavior to that claimed for it in the documentation, so that the user need not hesitate to follow the documentation exactly. In mechanical operation, keyboard actions having an irreversible effect on tape contents, such as erasing file entries, are denied automatic execution by being isolated from the formal command structure. The explicit system commands are all harmless in this respect, since none automatically replaces or deletes information, and a user can be comfortable guessing a command format or making typing errors. A command in execution, which may take several seconds, can be interrupted. Unrecognizable command statements are simply ignored.

Knowledge of the structure of LAP6, its files, indices, text strings, and tape layout, is not required for system operation. The structure has, however, been kept deliberately simple and is documented for the user for two reasons. First, it simplifies programmed interaction with the system. Second, the rare events of machine, tape, or program failure can never be entirely prevented. In an environment where it is possible to watch tapes move, however, the general transparency of system structure can turn normal response time delay into constructive feedback, reassuring the user and simplifying the problem of detecting real failure. Should this happen, explicit recovery procedures make it possible for the LAP6 user to recover fairly gracefully.

The LINC Users

The first LINC users, participants in the LINC Evaluation Program [4] sponsored by the National Institutes of Health,[2] were biological research professionals with little or no computer experience, selected specifically to evaluate the LINC as a tool in their particular research environments. LAP6 was written in partial fulfillment of the laboratory's commitment to this Evaluation Program group. Two months were spent canvassing these users and visiting their laboratories to observe both working environment and working habits.

The evaluators influenced both the number and functional characteristics of the system's commands, as well as the definition of its automatic filing capabilities. The editing facility, the program's most powerful feature, probably would not have been attempted without their stimulus.

Both the number of users (about 2000 in 1969) and their physical distance reinforce the several concerns with performance criteria already expressed. It is necessary for the LAP6 user to be able to understand and run the system on his own, using tape and documentation which arrive in the mail.

System Structure

LAP6 structure is determined solely by the allocation of tape blocks, as shown in Figure 1. The *current binary* area holds the user's program currently being checked out. A LAP6 text string held in the *scroll* area is identified as the *current manuscript*, which the user is currently preparing or editing. Unused blocks are reserved as "scratch area" for the user.

The Index and file of stored programs and manuscripts are optional on any tape, and they need not be on the LAP6 tape itself. Although only one tape is required for a complete system, it is usually more efficient to use two for file operations.

The size of the scroll area and the file area vary with different configurations of LAP6, the only limiting factor being the capacity of the tape.[3] The standard configuration shown has been found reasonable for most users.

Operation

The LAP6 user mounts his tape, executes a "read tape" instruction at the console, and pushes a start button. A display of the current manuscript appears. Thereafter the program is controlled from the keyboard. The user can type either manuscript or system *meta commands* (Figure 2). The manuscript, however, is the focus of LAP6, and manuscript manipulation is its normal "mode."

[2] In conjunction with the National Aeronautics and Space Administration under NIH contract PH 45-63-540.

[3] The actual tape locations of the various elements are assigned parametrically when the system is assembled.

Manuscript. A LAP6 manuscript is any useful collection of keyboard characters. There are no format restrictions, and the line length is virtually unlimited. LINC source programs are the most typical,[4] but applications include other programming languages [5, 6], flow charts [7], molecule descriptions [8], and bibliographies.

The current manuscript, continuously displayed, moves forward automatically as the user types. The line numbers on the left of the lines (Figure 2) are supplied by LAP6. The user can vary the number of lines displayed by rotating a potentiometer as he looks at the scope.

Scroll Editing

The user can look at, and change, any part of the current manuscript at any time. For example, in Figure 3(a) the manuscript is located at line 3132; the viewer indicates that he wants to see line 27 by typing "→27". He strikes the line terminator, and sees Figure 3(b). Perhaps he meant to replace line 25 with two new lines. He uses a special key combination twice to "back up" to Figure 3(c). He strikes the delete key, giving Figure 3(d), and types his two lines, Figure 3(e), which become lines 25 and 26. (Or perhaps he adds the two new lines at 3(c) before backing up and deleting line 25, or some combination. The order is unimportant.)

If the viewer now looks forward a few lines, he will see, Figure 3(f), his changes integrated in the surrounding manuscript and may notice that the former line 27, Figure 3(b), is now line 30. The former line 3131, where he began, is likewise now line 3132, which he will see the next time he looks at that part of the manuscript—that is, if he makes no more changes in between.

A few points suggested by the illustration can be illuminated:

1. THE SCROLL. The manuscript appears to move as a scroll, and is in fact treated by LAP6 as a scroll winding onto the two tape hubs. The unfurled part of the scroll, over the tape head, is displayed.

2. POSITIONING THE SCROLL. The viewer moves the scroll in either direction, displaying different portions. The motion is directed either by typing a line number (as "→27") or by striking one of four undisplayed key combinations (chosen to lie under the left hand) which move the scroll forward or backward, one frame or one line. The options simulate, rather crudely, forward and backward pushbuttons, with scroll motion stopping as the button is released. In the discrete situation the line numbers are usually used for gross adjustments and the key combinations for "fine tuning."

3. EDITING. The scroll is edited directly, as it moves across the scope. No distinction is made between input and editing, and there are no editing commands. The last line number displayed, e.g. "25" in Figure 3(d), identifies the viewer's *current line.* A line can always be added at the current line. The current line number is automatically incremented when the line being added is terminated. A line is deleted at the current line, or if "empty," the current line minus one. The current line number is appropriately decremented.

Less obvious from the illustration is the fact that the viewer can add or delete as much information as he likes wherever the scroll is located. Using the ADD MANUSCRIPT command, for example, a filed manuscript can be added to the scroll at the current line, amounting to inserting one manuscript in the middle of another.

4. LINE NUMBERS. The line numbers simply provide relative "you-are-here" information. They remain se-

[4] The acronym for *L*INC *A*ssembly *P*rogram is historical. The LAP6 language is adequate, but unremarkable.

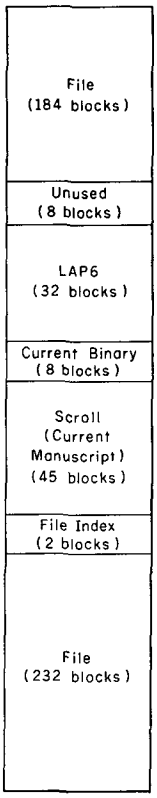


FIG. 1. LAP6 tape organization

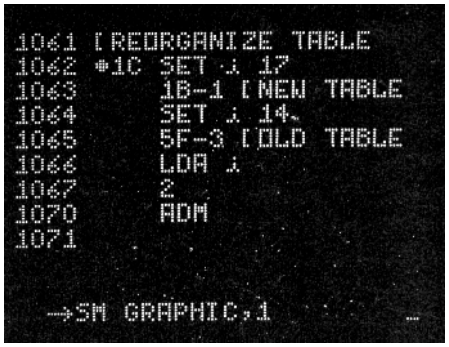
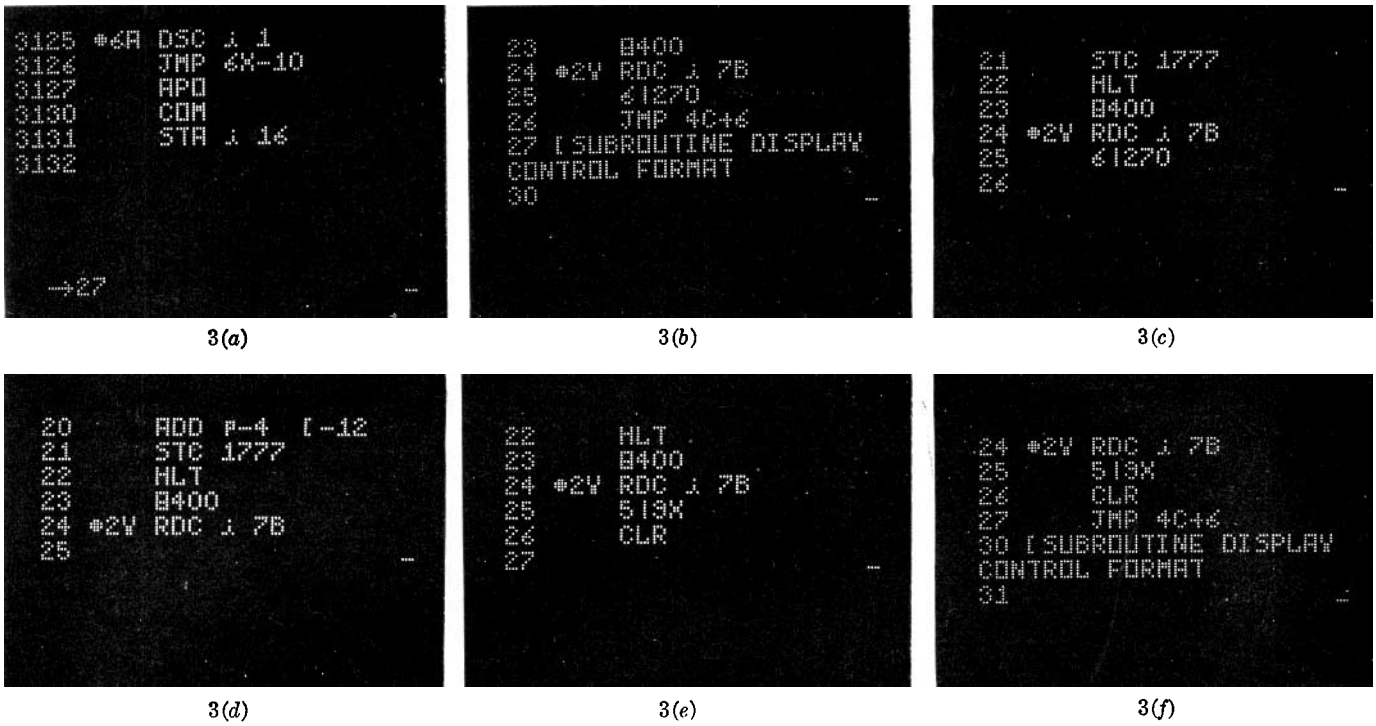


FIG. 2. Manuscript display



3(*a*) 3(*b*) 3(*c*)

3(*d*) 3(*e*) 3(*f*)

FIG. 3. Scroll editing

quential integers and are thus effectively renumbered by LAP6 every time a change is made. Although the viewer uses them to go to a line which is approximately where he wants to be, he always uses the "content vernier" provided by the display to insure that he is at the right spot. He is never required to know a line number in order to edit, nor to resequence the numbers.

From the standpoint of positioning the scroll the numbers are not essential, although some guide to the scroll's relative position is useful, especially with long manuscripts which may have no intrinsic order. The exact numbers, however, are more important in conjunction with some of the meta commands (Table I).

The fact that the line numbers change dynamically is popular with users. It focuses attention on the displayed content, and makes the editing process both faster and less error prone than schemes which permit blind editing with reference to arbitrarily assigned identifiers. The changing numbers also reassure the viewer; as he looks at subsequent portions of the scroll, he is frequently aware that the numbers are now higher or lower, confirming the effect of his earlier editorial corrections.

Manuscript Structure. A LAP6 manuscript is an unsegmented string of keyboard codes stored in consecutive tape blocks, in exactly the same order as they appear on the scope. This simple structure makes it easy for users either to generate or to employ manuscripts in other contexts.

The LINC Macro Expander is an example [5]. It uses a LAP6 manuscript of macro statements and definitions as input, and returns as output a LINC source program manuscript. Providing the macro language capability thus reduces to the problem of translating one manuscript into another. LAP6 is used to generate and edit the former, and assemble the latter. Either can be filed.

Editing Efficiency. LAP6 is used most frequently as a manuscript preparation tool, and efficient manuscript manipulation is critical. The LAP6 tape moves frequently, but briefly, during most editing activity. For the situation in which a number of changes must be made throughout a manuscript, the scroll editing technique is remarkably efficient.

Editorial perturbations to the scroll are localized, and their frequency minimized, by a compensatory deleting effect inherent in the editing algorithm [9, 10]; deleted characters compensate for added characters.

The tape motion which may result from editing near the boundary of a scroll block, being independent of the viewer's context, is sometimes surprising but not objectionable. Since the relevant part of the scroll is always over the tape head, there is no tape travel time loss, and the display is interrupted only for about 0.1 second, about the time it takes to strike a key.

Positioning the scroll, on the other hand, can introduce delays when the manuscript is long; the user quickly learns that sequential editing, although not required, is more efficient than random editing. To minimize these delays LAP6 uses a combination of techniques which move the scroll at either 23 blocks per second, or one block per second, depending on the status of the manuscript. The

23 to 1 saving is considerable when the viewer is simply moving the scroll and reading, as for example, when first finding his place in the manuscript.

It must also be remembered that when the user positions the scroll he *expects* the tape to move, and in fact watches it as it goes toward the requested portion. There is a psychological acceptance of the delay inherent in this cause and effect situation that is absent, for example, while editing, when only an immediate response is really tolerable.

In any case, the delays do not, in general, predominate. Since the burden is on the viewer to identify the context, he tends to spend as much time reading the scope as he does moving the tape, even when actively editing.

The standard 45 block scroll size (23,040 characters) has been found generous for most applications on the 2048-word LINC. No more than 2 blocks, however, are ever manipulated in the memory at one time, making the scroll editing technique attractive for small machine implementation. Longer manuscripts are equally efficiently handled if a device with faster transfer characteristics, such as a disk, is used to hold the scroll.

Meta Commands

The meta commands provide access to a basic set of operating features such as for manipulating file entries or loading programs. All use the scope or tapes, or both, in their execution.

The user states a meta command directly at any time, regardless of the position of the scroll or the extent of editing (Figure 2). It is executed when the line terminator is struck, and erased from the manuscript display to which LAP6 returns automatically. Execution of a meta command (except ADD MANUSCRIPT) does not affect the current manuscript.

Italics in the Table I summary identify command arguments. *LN* means a manuscript line number is to be supplied; *NAME*, the name of a file entry; *UNIT*, the number of the tape unit holding the file. Parentheses indicate optional arguments. For example, the command "→SM GRAPHIC,1" saves the current manuscript under the name GRAPHIC in the file on tape unit 1, whereas the command "→SM 2,603,GRAPHIC,1" saves only lines 2 through 603 of the current manuscript, etc. The command "→LO" loads the memory with the current binary program (the one last assembled), whereas "→LO GRAPHIC,1" loads a named, filed program.

Preparation of a LINC program illustrates command usage. The user enters his source program at the keyboard, doing any editing that may occur to him as he looks at his current manuscript. If his program is to run with, for example, a filed subroutine, he retrieves it with the ADD MANUSCRIPT command. LAP6 adds the subroutine manuscript to the current manuscript and returns to the manuscript display, now showing the end of the added manuscript on the scope.

The CONVERT command puts the binary version of the current manuscript onto the tape, and presents up to three displays showing: (1) symbol definition errors, if any, with their associated manuscript line numbers; (2) inclusive memory locations required by the binary program; (3) the symbol assignment table. The latter can be redisplayed at any time with the DISPLAY SYMBOLS command.

Single key options permit the viewer to step between these displays or return to the current manuscript display. If there are errors, he probably returns, corrects the errors, and converts the manuscript again.

The LOAD command transfers control to his program in the memory. The user generally leaves the LAP6 tape in place so that he can return quickly to look at the symbol table or to correct and reassemble the manuscript. The accessibility of the current manuscript has several implications. The "patch" technique has generally been eliminated from program check-out. "Binary" relocation is handled at the manuscript level, and the need for printed copy, although available, is greatly reduced.

The user can remove his tape at any time. When he returns to the computer and starts LAP6, his current manuscript, binary program, and symbol table are still intact. He eventually files the current manuscript or program ("→SM" or "→SB"). If there is already an entry with his requested name in the selected file, LAP6 protects it by displaying "REPLACE?" and requires him to strike a decision key before continuing. Otherwise LAP6 files the entry as directed and returns to the manuscript display.

The user leaves the computer by typing "→EX" and removing his tape. The EXIT command is necessary only if he still wants to retain the current manuscript.

Filing

Any file tape can be mounted on either tape unit. It is distinguished from a nonfile tape simply by the presence of an Index. Manipulating entries in tape files is straightforward, and combinations of meta commands are used when flexibility or structural control is desired.

Two file commands, SM and SB, move information from the LAP6 area on the tape into the file on either unit; three others, CM, CB, and CF, move information from the specified file to the file on the other unit, thus requiring two tapes. Since the LINC tape units are independently controlled, the position (block numbers) of the information being read from one tape is unrelated to the position in which it will be written on the other tape.

Index. A file tape contains a 2-block Index which can describe 126 file entries for the tape. The Index is created automatically when the first file command is issued to that tape, and erased from the tape when the last file entry is deleted. There are no commands for creating or erasing files or file indices.

An Index can be displayed or printed. The Index display (Figure 4) is responsive to the keyboard in a manner similar to the manuscript display in that it can be positioned

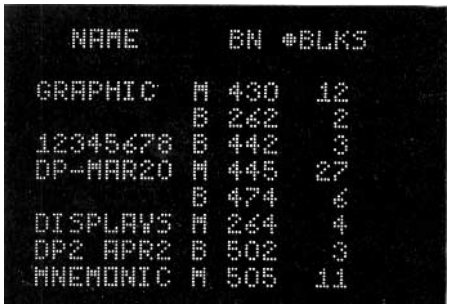


FIG. 4. LAP6 file index

in either direction and entries deleted with the delete key as the viewer reads. For protection LAP6 requires that the typist strike a decision key to make the deletions permanent. The "gaps" thus created between entries are automatically available to LAP6 for subsequent filing operations.

File Entries. A file entry is anything described in a file Index (Figure 4) as having a unique name with respect to others of its kind and as having some blocks (# BLKS) in the file area assigned to that name. LAP6 distinguishes manuscript entries (M) and binary program entries (B) as different kinds, which may, therefore, have the same name.

Filing Algorithm. File entries are variable and unrestricted in length, and the allocation of blocks within the file is freely formatted by LAP6 as entries are filed and deleted. The filing algorithm is influenced by the premise that a file Index is somewhere near the middle of the file area (so that no one entry can be more than half a file length away). It is, however, unaffected by LAP6 configurations for which this is not the case (specifically those with one-sided files), and is the same for all file commands: an entry is always saved in those contiguous, unassigned blocks which are (1) in the file area, (2) nearest the Index (either side, Figure 1), and (3) in which the entry will fit. The algorithm implies that, until entries are deleted, a file is always "packed" and its entries balanced around the Index, i.e. the two furthest entries are approximately equidistant from the Index.

Exercising the "replace" option thus does not mean that the namesake entry will physically occupy the same blocks as its ancestor. Tape travel time forces the simplicity of the algorithm, but its absolute predictability is sometimes useful. Inspection of the Index always shows where the next entry will be filed.

File Organization. Most of the time the user is not concerned with the file organization. He has, however, various ways of influencing it when either absolute or relative entry position is important. With a tape dependent system it is necessary to be able to reorganize files easily and quickly, perhaps moving certain entries closer to the operating system, or constructing a file with only binary programs, or simply balancing all present entries around the Index for more efficient general access.

The user can create the file Index himself as a LAP6 manuscript, allocating the absolute blocks of the file area in any convenient way, or by virtue of the filing algorithm, he can control the relative tape positions by the order in which he states file commands. To create an ordered file of binary programs, for example, from a mixed general file, or files, of programs and manuscripts, he has LAP6 execute successive COPY BINARY commands from the mixed file(s) probably into a new file, stating the commands in the order in which the programs are to be placed in the new file. LAP6 creates the Index if necessary, allocates the tape blocks for the new file according to the algorithm, and optimizes the tape motion of both tapes.

Copy File. Combinations of single entry file commands, such as COPY BINARY, and the COPY FILE command provide further flexibility when organizing files. COPY FILE copies all entries from some "File I" into some "File II," except for duplicate entries already present in File II. Thus, following single entry filing operations, COPY FILE can be thought of as copying all the "remaining" entries.

Tape efficiency is optimized as for the other commands. Entries transferred into File II are packed and balanced; entries on one side of the File I Index may therefore appear on the other side in File II. In addition, both tapes are handled with the fewest "search reversals," i.e. the fewest changes of direction of motion due to searching. This means that the File I tape is entirely read with two reversals, and the File II tape entirely written and checked with four, regardless of the number or position of the relevant blocks or gaps on either tape. Given this type of tape, no faster method of tape file copying exists [11].

Layering

Segmenting programs into layers, which are read into the memory from the tape only as needed, is necessarily common LINC practice. In the LAP6 case layering has some advantages, and the time required to change layers, about 1 second, is negligible in the on-line situation.

LAP6 is started (or reentered) with the "manuscript display layer" which handles keyboard input and display of manuscript and meta command statements, all scroll editing activities, and decoding and dispatching of layered meta commands. Most other layers return directly to this layer without further layering activity.

LAP6 has no resident control program. When the layer currently in control calls the next layer from the tape it makes the entire memory available to it. User designed features, or programs which interact with the system, likewise have no restrictions placed on the use of the memory.

Layering Tape Instruction. Layering is greatly simplified by the power of the LINC's tape instructions which can be used the way a resident operating system uses "jump" instructions to transfer control between major segments of the program. This is perhaps best illustrated

when the active layer must read over itself, as when the incoming layer occupies the entire memory. The layering "routine" consists of one read tape instruction. When this instruction is executed, the computer gets the next instruction as usual but takes it from the incoming layer.

Although this fixes the layer's entry point arbitrarily, it standardizes the linkage and makes communication between user programs, or new layers, and the operating system a trivial problem.[5] Features are easily added without having to modify LAP6. The FREE meta command provides layered access to an unused block within LAP6, which the user can assign to a meta command of his own design. Except for its proximity to LAP6, the access is the same as for any user program which LAP6 can LOAD from a file.

A user can ignore the layering conventions and reenter LAP6 at any time from the console, recovering both current manuscript and binary program. His program, however, can also use its own tape location (supplied by LAP6 during loading) to "layer into" a completely different system, which can subsequently reenter LAP6 under program control automatically by executing the layering tape instruction.

Read-only Layering. Since write and check operations on the LINC are time-consuming [12], LAP6 layering is usually a process of destructive read operations; the active layer generally does not save itself on the tape before reading the next layer.

This "read-only layering" has the advantage of conserving memory as well as tape time, since reading the tape amounts to "presetting." It is estimated that LAP6 would need an average of 30–35 percent more core memory per layer if it had to accommodate full presetting operations and generated tables. As it is, most layers are not only self-modifying but also generate tables destructively on top of finished portions of the layer. The COPY FILE command layer, for example, reduces its incoming 600+ instructions to 100+ when active.

Conclusions

LAP6 has been in use on LINCs, LINC-8s, and micro-LINCs since the summer of 1967. Its success can be at least partially attributed to having given top priority to the general criteria of efficiency, reliability, and operational simplicity as they are understood in the on-line environment. The small memory was found to influence functional specifications less than other considerations and perhaps operated with a positive effect on the criterion of simplicity. Compromises were, of course, necessary, but we also found that operating features which may seem highly desirable, for example, to a professional in the computer field, can be so much excess baggage in an on-line applications environment.

[5] To minimize the inconvenience, special memory "index" registers, which on the LINC are generally preset anyway, are used to hold the linkage.

The system's most popular features are its editing facility, its straightforward command and filing structure, and its documentation. In addition to the operating instructions, the forty page *LAP6 Handbook* [3] includes a description of the LINC programming language, notes on structure and LAP6 configuration, and recovery procedures in case of machine or tape failure.

The system was used for 6 months by 15 colleagues at Washington University before becoming generally available. The period was adequate to establish reasonable reliability, and this group also contributed several simplifying functional changes.

The preparation of LAP6 involved 5 months for specifying and general planning, 2 for documenting, and 13 for algorithm study, programming, and check-out. It has approximately 4600 instructions. Our experience with it has demonstrated the feasibility of providing a reliable and efficient on-line operating system, useable in a variety of environments, and running on a very small computer.

Acknowledgment. The author is grateful to Dr. Jerome R. Cox, Jr., for several helpful suggestions.

RECEIVED JANUARY, 1970; REVISED MARCH, 1970

REFERENCES

1. CLARK, W. A., AND MOLNAR, C. A description of the LINC. In *Computers in Biomedical Research, Vol. 2*, R. W. Stacy, and B. Waxman (Eds.), Academic Press, New York, 1965, pp. 35–66.
2. BEST, R. L., AND STOCKEBRAND, T. C. A computer-integrated rapid-access magnetic tape system with fixed address. Proc. Western Joint Comput. Conf., Vol. 13, 1958, Spartan Books, New York, pp. 42–46.
3. WILKES, M. A. LAP6 handbook. Tech. Rep. No. 2, Computer Research Lab. Washington U., St. Louis, Mo., May 1, 1967.
4. Convocation on the Mississippi, Proc. Final LINC Evaluation Program Meeting, Washington U., St. Louis, Mo., Mar. 18–19, 1965.
5. FRANKFORD, C., AND ELLIS, R. A. LIME (LINC Macro Expander). Tech. Memo. No. 3, Computer Systems Lab. Washington U., St. Louis, Mo., Aug. 24, 1966.
6. MACK, H. L. An introduction to BLS: the BASIC language system. Tech. Memo. No. 95, Computer Systems Lab. Washington U., St. Louis, Mo., Mar. 1970.
7. GRAESSER, S., AND ELLIS, R. A. Direct utilization of flowcharts to represent macromodular systems. Computer Systems Lab. Tech. Memo. No. 60, Washington U., St. Louis, Mo., Apr. 1968.
8. ELLIS, R. A., GRAESSER, S. M., BARRY, C. D., AND MARSHALL, G. R. MOLGRAPH: a program to manipulate and display molecular models. Tech. Memo. No. 86, Computer Systems Lab. Washington U., St. Louis, Mo., July 1969.
9. WILKES, M. A. LAP5: LINC assembly program. Proc. DECUS Spring Symp., May 1966, Digital Equipment Corp., Maynard, Mass., pp. 43–50.
10. ——. Scroll editing: An on-line algorithm for manipulating long character strings. *IEEE Trans. Computers* (to be published).
11. ——. An algorithm for fast tape file copying. Computer Systems Lab. LINC Doc. No. 76, Washington U., St. Louis, Mo., Feb. 1970.
12. ——, AND CLARK, W. A. Programming the LINC, 2nd ed. Computer Systems Lab., Washington U., St. Louis, Mo., Jan. 1969.