COMPUTER TECHNOLOGY

Pleiades: Real-time-sharing control in the behavioral science laboratory*

JOE L. LEWIS†
University of Washington, Seattle, Washington 98192

and

G. W. OSGOOD and J. J. HEBERT
University of Oregon, Eugene, Oregon 97403

Pleiades, a seven-user, real-time-sharing executive was developed on a PDP-15 computer for control of psychological experiments. The goal of the system development was to provide millisecond timing accuracy for all users and to permit easy computer utilization for the novice user. Two modes of operation are available: the standard language approach and the experiment writer approach. The system is discussed from the user's viewpoint, and a brief description of the details of system design is given. The system has been operating successfully for several months.

The availability of computers in the behavioral science laboratory has increased considerably in the past few years. The technological achievements in computer hardware have been remarkable. Unfortunately, the software development required to permit easy implementation of computer control in the laboratory environment has not kept pace with the hardware development. Consequently, psychologists who do not wish to become computer programmers may resist using a computer even when it is available. A desirable goal would be to provide sophisticated computer capability to the novice programmer or nonprogrammer without requiring that he undergo a long training period. The problem is to provide an interface between the language of the psychologist and that of the computer. There are several possible approaches: (1) Use a standard computer language such as FORTRAN. Standard subroutines must be supplied to perform special-purpose functions or control special devices (see Lewis, Boies, & Osgood, 1971). (2) Provide a special language that is easily learned by the E. Examples of these are PSYCHOL (McLean, 1969), developed at Carnegie-Mellon, and SCAT (Grason-Stadler, 1968), which is currently being extended at the University of Colorado (Polson, 1971). (3) Provide the computer with a subset of the E's language, so that the E can communicate more or less directly with the machine. This approach requires that the computer interpret the E's language rather than that the E learn to use the computer's language. It is similar to the second approach, except that it attempts to include experimental design language as well as English.

The particular computer configuration (both hardware and software) that is desirable depends on the particular usage anticipated. Some of the important considerations include: (a) cost of hardware and software development, (b) time (delay) required to fully implement the system, (c) range of experiments that the researcher wishes to control with the computer, and (d) range of the user's computer skills.

If the number of users is small and the experimental paradigm is relatively constant, one or more small dedicated application computers may be desirable. This is particularly true now, with the low cost of central processors and memory. If the number of users is large and the variety of experimental procedues is great, an in-house general-purpose time-sharing system may be desirable. The purpose of the present paper is to describe one implementation of the latter approach. The general-purpose real-time-sharing system we describe here is operating on a PDP-15 computer at the Center for Cognitive and Perceptual Studies at the University of Oregon.

GOALS OF SYSTEM DEVELOPMENT

The goals established for development of the system were as follows: (1) The time-sharing part of the system was to be designed so that the user would be unable to detect that he did not have complete control of the machine; timing accuracy had to be maintained, displays had to remain constant, and errors in one user's program could not interfere with other users or "crash" the system. (2) Sufficient space and system support would have to be available for each user, so that the program development, debugging, and execution of real-time programs could be possible in the time-sharing mode without requiring a dedicated system. (3) The novice programmer and the nonprogrammer would have to have reasonable access to the computer capabilities without extensive training.

The initial version of the system was written to accommodate two users. Only minor modification would be required (primarily changing constants) to

*This research was supported by the Advanced Research Projects Agency of the United States Department of Defense, monitored by the United States Air Force Office of Scientific Research under Contract F44640-67-C-0099 (while the senior author was at the University of Oregon), and by the United States Air Force Office of Scientific Research, Air Systems Command, under Grant 70-1944, to the University of Washington.

†The authors wish to acknowledge the contributions of Stephen Boies, Manard Stewart, and Nancy Frost. Without their many ideas, contributed through many hours of discussion, the system would not be at its present state of development. Allen Murphy was responsible for the technical details of the hardware design and modification.

expand the system to a maximum of seven users.

The PDP-15 computer on which the system is implemented has 12k of 18-bit words of memory, with an 800-nsec cycle time. The central processing unit (CPU) contains hardware arithmetic, a real-time clock, and a modified memory protection containing two protection bounds and an instruction trap. Interaction with the computer is through one of three Teletypes, the high-speed paper-tape reader-punch, or the three-drive DECtape mass-storage system. Both the software and the peripherals can be logically divided into components dedicated to the system and to components dedicated to each user.

The system components consist of one 4k page of core memory, in which the system executive and re-entrant system routines reside, the computer console, a console Teletype, and one of the DECtape drives (there is no disk). Each user's components consist of one 4k page of memory, one DECtape drive, a Teletype, and a S station. Each S station includes a CRT display, a 10-bit input switching network accepting keyboard or relay closure information, and a 12-bit output switching network for manipulating external devices (e.g., tape recorders, lights, etc.). In addition, a white-noise generator, a waveform generator, and the paper-tape reader-punch are available in common for the users.

A larger number of re-entrant device handlers and special experiment control subroutines reside in the system page. The handlers permit asynchronous input/output (I/O) on all of the peripherals. The subroutines provide special functions, such as randomization, timing of events, recording of response times, plotting on a CRT, control of special-purpose apparatus, as well as dynamic space allocation and task scheduling.

USER-SYSTEM INTERACTION

The user initiates interaction with the system by typing control characters on his Teletype. (The control key is represented by ↑.) The system page contains a permanently resident command interpreter that decodes the control characters and initiates transfer of appropriate programs from the system DECtape into the user's 4k page of memory. Two approaches are implemented to permit the user to set up a computer-controlled experiment. Both are intended to require minimal training in computer science. The first to be described requires about 20-30 h of training for a novice; only a few hours are required if the user has programming experience.

STANDARD LANGUAGE APPROACH

The standard language approach requires the user to learn a subset of FORTRAN or a subset of PDP-15 assembly language to control the sequence of events and to call the special subroutines available in the system page. Most of the controlling program consists of calls to subroutines with the associated arguments. Therefore, a substantial part of the novice's training is spent becoming familiar with the operation of available routines. Useful subroutines include graphics control, response and event timing, and task scheduling. For example, a FORTRAN WRITE statement can be used to display any information on the CRT that could be written on a Teletype or line pringter using the WRITE. However, the CRT is not restricted by type size or character set. Available character sets include both upper- and lowercase letters, numbers, Hebrew letters, Gibson figures, special characters, and user-defined character sets. Four type sizes are available and four levels of gray (brightness) can be used with any CRT display. A CALL statement can be used to plot any figure that can be constructed by using points. Response timing can also be initiated with a CALL statement. The S's keypress causes the reaction time (RT) to be passed to the user's argument list. Finally, task scheduling is carried out by using the subroutine DEFER. DEFER requires two arguments: the time delay in milliseconds and a subroutine name. The CALL to DEFER does not cause immediate transfer of control to the named subroutine. It retrieves the arguments and returns control immediately to the next executable statement in the user's program. After the delay, control (along with arguments) is passed to the named subroutine. RETURN from the subroutine returns control to the user at the location being executed at the end of the delay. DEFER is a re-entrant routine, so that multiple CALLS with different delays and subroutine names may be initiated in rapid succession. This technique permits easy scheduling of multiple events, all of which are timed relative to some reference time (e.g., the beginning of the trial). The power of the subroutine set may be illustrated by noting that only a few statements are required to display a geometric figure, to rotate it in real time, to display a letter of the alphabet beside it at some randomly determined rotation, and to record the S's RT to the letter. Only a few additional statements would be necessary to monitor the S's simultaneous tracking of the figure with his other hand. Because users' programs are usually not very large, the 4k user page of memory provides plenty of space. In fact, users can generally run larger programs with the 4k page than they can using vendor-supplied software in an 8k dedicated machine.

The general procedure using standard language is to create the program in a file on DECtape, using a text editor. The user must then assemble or compile the program, load it, and debug. Program development normally begins by typing ↑E on the Teletype. The executive will respond by transferring the text editor into the user's 4k page. When the editor is loaded, it signals its readiness to accept commands by typing EDITOR on the Teletype. A variety of commands are recognized. Files may be created, modified, and searched. All of the vendor-supplied PDP-15 editor line mode commands are accepted as well as some additional features, such as multiple search functions. After the

program is created on DECtape, the user calls for the assembler[1] or FORTRAN compiler. The assembler or compiler accepts the file created by the editor and generates the appropriate relocatable binary code in another file on the user's DECtape. If errors are detected, it returns to the editor for modification. Otherwise, the linking loader is requested. The linking loader loads the user's binary program from his DECtape. It then searches the DECtape for local subroutines and loads them as needed. Finally, it searches the library file on the system DECtape for the remaining required routines. The system library contains the routines necessary to link to the permanently resident special subroutines as well as the infrequently used subroutines (e.g., some floating point arithmetic functions).

The standard language approach permits maximum power in control of experiments. All of the system capabilities are available to the user. Its disadvantages include the unavailability of the FORTRAN compiler in the time-shared mode and the relatively long training (20-30 h) necessary for the novice.

THE EXPERIMENT WRITER APPROACH

The experiment writer is an attempt to provide the computer with the language of the E. It permits a very large subset of possible experiments using the standard language approach, and requires only about 2 h of training.

The basic approach is to define stimulus and response sets and to specify the procedure for trial and block definition. The constraints, of course, are that stimulus and response files, trial definition, and block definition must be specifiable by using the procedures available in the experiment writer.

FILE DEFINITION

Files are created on the DECtape by using the editor. The experiment writer is graphics oriented so that the stimulus file is frequently a definition of characters or patterns that will be displayed on the CRT. All of the character sets available using the standard language approach are available to the experiment writer. The stimulus file may be a list of letters, words, or sentences. They may be in a variety of type sizes or character sets. They may be positioned anywhere on the CRT screen and may be displayed in four levels of gray. Patterns may also be defined as a stimulus file. Currently, they are defined by specifying coordinates for the points to be displayed. Vector-generation software is under development to permit the specification of key points in a display along with a function to be used to connect them.

Response files may also be created for use by the experiment writer. The response file is used to link the appropriate response with each stimulus in order to determine if S's response is correct. The first type of response file defines the correct response for a given trial type. The second uses the stimulus number in the stimulus file as an index to locate the corresponding response item in the response file. Feedback concerning the correctness of the response, as well as the response latency, is thus made available for the S. In addition, incorrect responses may be identified and deleted in calculating the summary statistics that are available at the end of each block of trials. Special apparatus, as available with the standard language approach, may also be used. However, they do not usually require file definition.

TRIAL DEFINITION

A trial is defined by using a combination of 10 commands (key words) recognized by the experiment writer. The commands fall into three categories: display onset and offset, timing, and conditional.

The display onset commands are SHOW, ON, and FEED. SHOW, followed by a file name, instructs the experiment writer to display an item from a file on DECtape specified by the name. Special characters typed after the file name indicate whether the items in the file should be presented in order or permuted randomly. Special characters are also used to indicate the type of linkage that should be made with response files and to indicate whether the stimulus item should be saved in the data file. The second command, ON, is used to turn on apparatus connected to the output switching network. Any apparatus can be used that can be controlled by switch closures. The FEED command is used to display the S's reaction time and a + or −, indicating whether the response was correct or not. The display offset commands are DELETE and OFF. DELETE is used to turn off CRT displays; OFF turns off the switches.

There are three timing commands, one for intervals and two for responses. The WAIT FOR or FOR command inserts a delay in the trial. It requires one argument indicating the time in milliseconds to delay. The next event specified does not occur until the time has elapsed. The TILL command is also used to insert a delay in the trial. However, the argument indicates a keypress to end the delay. The ALLOW command is used in combination with a display onset or offset command to start timing the S's reaction time. One argument specifies which keys are enabled for the response.

The remaining command, WHEN, permits response-contingent action. It has one argument indicating the response expected. It also accepts any of the display onset or offset commands. When the response is detected, the display command is executed. It is useful for specifying a secondary task in the trial definition.

In general, the arguments associated with the above commands may be constants or literals, file names, or variable names. If a variable name is used, it is defined at run time. Thus, the same experiment may be run under a

variety of parameter specifications.

An example will be given to illustrate the trial definition. The example defines a "different" trial in the Posner (Posner et al, 1969) letter-matching experiment.

```
SHOW LET!S      ; Show a randomly drawn (without
                  replacement) letter from the file
                  LET.
WAIT FOR MSEC   ; Insert a delay in the trial. MSEC must
                  be defined at run time.
SHOW LET!S      / Show a second letter from the file
                  LET.
ALLOW 1-2       / Start timing S's RT.
WAIT TILL 1-2S* / Wait for S's keypress.
DELETE LET      / Delete both letters when the response
                  is made.
FEED DIF FOR 1000 / Display the S's RT for 1 sec.
WAIT FOR ITI    ; Wait for an interval to be defined at
                  run time.
```

A letter is displayed for MSEC milliseconds until it is joined by a second letter. Both go off when the S responds. The RT is displayed on the CRT for 1 sec followed by an intertrial interval defined by ITI. The stimulus numbers, the RT, and the response number are stored on tape for later analysis.

A given experiment may have only one trial type or it may have several. A particular type of trial will occur as specified in the block definition.

BLOCK DEFINITION

The block definition establishes the sequencing of trial types. The simplest case is the one-trial type. The block definition simply designates the number of trials per block. Several trial types may be used in a block, and they may be arranged in a variety of ways. An E may wish to have 10 learning trials followed by a single test trial to define a block, or he may wish to alternate trial types. The foregoing are all illustrations of fixed trial order.

Permuted trial sequences may also be defined. For example, 20 "different" trials (as illustrated in the previous trial definition) may be mixed randomly with 20 "same" trials to form a block of 40 trials. Of course, restrictions may be imposed on the randomization. For example, one may define a block of 40 trials as four sets of 5 "same" trials mixed with 5 "different" trials. The block definition is not restrictive: 10 or more trial types could be used in a variety of combinations.

Several data-handling features of the experiment writer are available. (The RT feedback has already been mentioned.) Raw data is normally output onto DECtape for later analysis. Special characters in the trial definitions are used to mark data or stimulus information for permanent storage. In addition, summary statistics are available at the user's Teletype at the end of each trial block. Consequently, one may examine errors or RTs for a given trial type at the end of each block to determine parameter values to be used for the next block.

GENERAL OPERATION OF THE EXPERIMENT WRITER

To create an experiment, both the editor and the experiment writer are used. The editor is used to establish stimulus and response files and, if desired, trial definition files. The experiment writer is used to establish trial definition (or retrieve trial definition files) and for block definition. It is designed to permit interim output as well as the completed experiment to be put onto DECtape for permanent storage. The experiment writer is set up with a full set of error-handling routines to inform the user of his mistakes and to attempt to restore control to a point just prior to the error. If interim output is stored, it is not necessary to start from the beginning in case of an error. When the experiment is set up, the experiment writer generates executable machine code and establishes links with the subroutines in the system page. The binary code is then dumped onto DECtape for immediate use at run time. To run the experiment, one need only retrieve the page from DECtape, type in values for variable names, provide starting numbers for any random number operations, and specify the number of blocks to be run.

Activating an experiment usually takes only an hour or two. Modifications to an existing experiment can often be made in a few minutes. Not all studies that can be generated using standard language can be done with the experiment writer. For example, if the nature of a trial depends on the S's response to a previous trial, it cannot be used. And sophisticated rule definition procedures are unavailable. Nonetheless, the experiment writer represents the quickest way for the novice to make productive use of the computer. For many experiments, the experiment writer is the best tool, regardless of the programmer's sophistication.

EXECUTIVE FUNCTIONS AVAILABLE

The user may execute and debug his program under the real-time-sharing executive. With the program loaded into the user's page, one may search for particular instructions or address references, examine desired locations, and alter those locations. One can insert breakpoints to monitor the program flow and restart the program after modification to try the run again. Changes can be made permanent by dumping the corrected version onto DECtape for retrieval at run time.

The executive also has several DECtape file-management capabilities. The DECtape directory can be listed or cleared; files on the DECtape can be renamed, deleted, or listed on the Teletype. Blocks of information on DECtape can be accessed directly (without reference to the file name) for listing on the Teletype or for direct permanent modification.

REAL-TIME-SHARING SYSTEM DESIGN

Both the hardware and software designs were important in the successful implementation of the time-sharing system. The basis for the hardware design

has been presented elsewhere (see Boies, Lewis, & Murphy, 1970); an example is the use of a direct memory access-type CRT controller. With this type of controller, it is necessary only for the CPU to start the plotting function. Otherwise, all plotting goes on asynchronously, requiring very little system overhead to maintain displays.

The system provides substantial user capability and can be implemented within a reasonable time period. It is not, however, the ultimate in computer science sophistication. Basic to the design is a fixed core partition (4k) for the system and for each user. The executive allocates a 2-msec time slice, rotating control to each user. Consequently, if the system is expanded to its maximum capacity (seven users), each user would have a maximum delay of 12 msec between time slices. However, the actual system operation is such that a user would rarely wait for the maximum delay. Since the user's program is largely a series of CALLS to subroutines for timing and event manipulation, the computer is usually executing system routines.

The system is designed so that all input and output is asynchronous. Whenever one user is waiting for an event (probably 95% of the time in a typical experiment), the system automatically passes control to the next user in rotation. In addition, all system routines are written in such a way as to guarantee high-priority interrupt handling within about 100 μsec. Interval timing for all events can be accomplished by using the DEFER subroutine. With DEFER, interval timing is also done at the interrupt level and millisecond accuracy for intervals can be guaranteed. The 2-msec time slice may seem small in light of the larger intervals usually used in time-sharing systems. However, in an experimental control and program development environment, as opposed to "number crunching", 2 msec is sufficient for a large percentage of the typical user's job. It requires about 120 μsec to perform the swapping, or 6% system overhead for a 2-msec time slice.

The asynchronous I/O management is facilitated by two hardware features: a four-level priority interrupt system and a memory protection, instruction trap facility. The priority interrupt system permits high-priority devices (e.g., the DECtape or response system) to interrupt the handling of lower priority devices (e.g., Teletype). Low-priority devices wait in a queue until high-priority devices are finished. The memory protection facility provides a high and low boundary[2] that can be set under executive control. It is used to limit the memory that is addressable by each user. The system and users are protected from each other. If a user tries to read and write in memory outside of his own page, control is automatically passed to an executive routine which checks the legality of the memory access. Control is also passed to the executive if a user tries to use certain restricted instructions such as I/O transfers or a system halt.

Several other features have been included in the system design to increase throughput and reduce user interaction. Space required for temporary storage, such as for I/O buffers (e.g., to store points for CRT plotting), are dynamically allocated by the system. Thus, sequential tasks do not require separate array storage areas. The CRT displays are refreshed asynchronously by using a direct memory access controller. The refresh is also clocked at a constant interval rather than carried out continuously. Therefore, variations in the system load do not influence the brightness of the CRTs.

The system has one potential bottleneck. There is only one controller for all DECtape drives. If one DECtape drive is in use, then the others are locked out until the current block transfer is complete. The DECtape handler was written to use a dynamic multiple buffering scheme to insure that data are not lost or that the user is not delayed by the DECtape hardware. Modification of the handler is under way to incorporate a priority-handling system. Users performing low-priority tasks (e.g., program development) will be delayed at the expense of high-priority tasks (e.g., experiment control).

SYSTEM PERFORMANCE

At the present time, the computer operates under the time-sharing executive about 8 h per day. It operates under a single-user system during the remaining time. The percentage of time under the time-sharing executive should increase when the assembler is finished.

In general, errors made by a user are handled by the executive without affecting the other users or the integrity of the system. When an error is detected, it is analyzed and one of more than 40 error messages is printed on the user's Teletype. A special hardware diagnostic routine is also available to each user. He may load the test routine into his page and quickly check the operation of his CRT, input switches, and output switches. Debugging time can be minimized when, in fact, there is a hardware failure. The ideal goal of system reliability is to never have a failure which cannot be explained. Unfortunately, that goal has not been reached. Currently, there is about one unexplained system "crash" about every 2 weeks.

It has been claimed that the system should provide millisecond timing accuracy for all users. Fortunately, the actual system performance can be monitored. The PDP-15 is connected through a direct link (interprocesser buffer) to a PDP-9 computer in the same laboratory. The operation of the PDP-15 has been monitored by the PDP-9, using a variety of job mixes. Timing error for both intervals and responses has in every case been less than 1 msec. It must be noted that the system is currently operating with only two users. However, in making the evaluation, no attempt was made to guarantee timing accuracy (e.g., by using only programs which use DEFER for task scheduling).

The system will soon be expanded by two additional

users. The additional memory, Teletypes, DECtape drives, etc., have already been ordered. When the hardware is installed, the system conversion to four users should require only a few hours of work. Evaluation of the expanded system using the PDP-9 for monitoring will be continued.

The development of the system to its present state was done over a period of about 22 months from the time of the computer delivery. However, the programming was carried out on a part-time basis. Most of the computer's time during the period of development was devoted to single-user experiment control. The actual development time is estimated at 9 man-months for the time-sharing executive and system subroutines and 4 man-months for the experiment writer. None of the vendor-supplied software was used intact, except the FORTRAN library routines. All the handlers, the editor, loader, etc., were written from scratch. It is definitely a major programming effort to write a time-sharing system with the capabilities that were desired. However, we feel that the end product is well worth the effort and the time expended. It should also be noted that the development time compares more than favorably with other approaches that have been completed or are under development elsewhere.

The system development is an ongoing project. As noted, the assembler and vector routines are currently under development; a FORTRAN compiler to create assembler code is currently under study. The system has not been evaluated sufficiently at present to indicate where it should be modified or expanded.[3]

REFERENCES

Boies, S. J., Lewis, J. L., & Murphy, A. R. Hardware design for a multiprogramming system for a medium scale computer. In Decus Proceedings, Spring 1970.

Grason-Stadler *The SCAT primer*. West Concord, Massachusetts: Grason-Stadler Company, 1968.

Lewis, J. L., Boies, S. J., & Osgood, G. W. Zoroaster: A multiprogramming system for psychological research. Behavior Research Methods & Instrumentation, 1971, 3, 106-107.

McLean, R. S. Psychol: A computer language for experimentation. Behavior Research Methods & Instrumentation, 1969, 1, 323-328.

Polson, P. G. Extended state algorithm translator reference manual. CLIPR Publication Series, No. 2, University of Colorado, Boulder, 1971.

Posner, M. I., Boies, S. J., Eichelman, W. H., & Taylor, R. L. Retention of visual and name codes of single letters. Journal of Experimental Psychology Monograph, 1969, 79, 1-16.

NOTES

1. At the time of this writing, the assembler is unavailable in the time-shared mode of operation. It should be completed before this paper appears in print. The FORTRAN compiler must be used on the dedicated computer. A 4k compiler is not scheduled at the present time. It is unclear that the payoff would be worth the work involved.

2. The standard memory protection available for the PDP-15 provides only one boundary check. The hardware was modified to provide the second boundary.

3. Copies of the software and documentation of all hardware modifications are available from the authors.

(Received for publication January 19, 1973;
revision received March 29, 1973.)