THE SDC TIME-SHARING SYSTEM

facilities, services & potential

by JULES I. SCHWARTZ

In spite of all the power available in present day computers, a feeling of uneasiness, even disappointment seems to prevail among those concerned with paying for and using computers. These feelings stem primarily from the relative inaccessibility and inflexibility of the computer. The ability of users to produce computer systems that can be readily modified is frequently much less than supposed, and the ability of computers to afford quick and accurate solutions to a variety of even simple problems is surprisingly limited. Part of the reason for this dilemma is the formidable wall between users and computers in most installations. This wall is faced not only by managers and customers, but also by programmers.

One of the major reasons for the wall is the persistence of traditional techniques for applying computers. These techniques require that programs be prepared (in a language that is not usually oriented to application), be sent to the computer (where they are processed and executed with no intervention by the user), and, when processed, be sent back hours (or days) later. Aside from the fact that this approach can lead to numerous delays and wasted computer runs, there are many applications for which this detached operation is completely unsatisfactory. Thus the concept of continuous interaction by the user with his computer (*on-line* computer operation) promises to become an essential part of current techniques in computer application.

On-line use of a computer is not a new concept, for it has appeared in a variety of situations in the past; but these applications have been special purpose (e.g., SAGE) or quite limited (e.g., use of small computers such as the Bendix G-15 or the Digital Equipment Corporation's PDP-1). Complete user interaction may be available on small computers, but because of their properties, many desirable features of large on-line systems cannot be made available, such as access to many general- and special-purpose languages. Therefore, it seems, use of a large computer on-line would be desirable. Unfortunately, such use would introduce certain inefficiencies, without major changes in technique. For one person to preempt the total capacity of a large machine for long periods of time would be highly uneconomical if he could not keep the computer occupied all the time it was assigned to him. Time-sharing permits on-line use of the computer simultaneously by a large number of people by giving each user time when he requires it. This kind of system provides a direct and continuous working relationship between users and computer and keeps the computer busy most of the time by limiting the amount of idle time due to human thought or output from on-line devices.

In the Command Research Laboratory at SDC, a large percentage of the problems run are of the on-line, man-machine, interactive variety. Since these applications are virtually impossible to run in a serial fashion (one-at-a-time), the requirement to produce a system that would permit parallel running was a necessity. For this, and for the sake of further study of the time-sharing process itself, a time-sharing system was developed as the main program vehicle with which to run the laboratory.

characteristics of time-sharing systems

Given the general requirements for time-sharing systems, it may now be worthwhile to examine the properties (and hence the definition) of such systems. Four characteristics of time-sharing systems encompass most of their distinctive features; such a system is:

- Simultaneous—A number of people use the computer at the same time.
- Instantaneous—All users receive responses from their

Mr. Schwartz is head of the time-sharing project at System Development Corp., Santa Monica, Calif., with whom he has been associated since prior to its spin-off from RAND. His software development projects have included PACT, the Lincoln utility system, and JOVIAL (Jules' Own Version of the International Algebraic Language). He holds a BS in math from Rutgers and an MA in mathematical statistics from Columbia Univ.

programs or the system within seconds—or fractions of a second—of the completed computation.

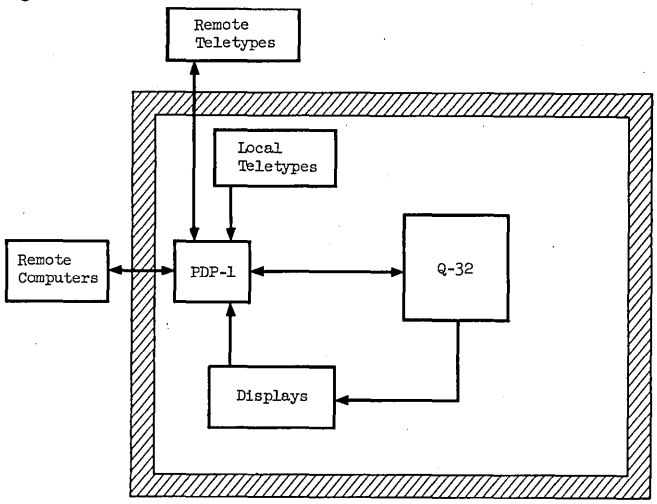
- Independent—Different programs, services, and devices can be in use separately or in combination during any given period of time.
- General-purpose—No restriction is placed on the kind of program or the application under the time-sharing system.

Various on-line systems have some, but not all of the properties just described. Others, such as the SDC system, do have all of these characteristics.

equipment configuration of the SDC system

Since on-line operation of a computer requires the use of input-output devices in addition to tapes, card-readers, printers, etc., on-line devices available to users of the system will be reviewed first. These devices include teletypes, typewriters, and cathode-ray tube (CRT) display

Fig. 2



consoles. The teletypes have been installed in various locations within SDC and are also used from numerous remote locations, as far away as Pittsburgh, Pa. Within SDC, there are eight model 28 Teletypes, 16 model 33's and three Soroban typewriters. There also exists capacity for eight remote teletype stations operating simultaneously. In addition to three types of keyboard devices, there are six display consoles currently available. These are all located within the Command Research Laboratory.

The SDC time-sharing system may also be used remotely via a 2kc line, the terminal to which may be a computer. That computer may in turn service displays and keyboards. This linkage permits the remote computer to communicate with a program in the SDC computer, thus permitting computer-computer-human interaction under time-sharing. Currently one such line is operating between the AN/FSQ-32 in Santa Monica and a CDC 160-A computer at the Stanford Research Institute, Palo Alto, Calif.

The basic characteristics of the time-shared AN/FSQ-32 system are listed in Table I. Assisting the Q-32 under time-sharing is the PDP-1, which serves as the major interface between the on-line devices and the Q-32. A simplified diagram of the complex is shown in Fig. 2. The local Q-32 configuration is shown in Fig. 3.

Fig. 3

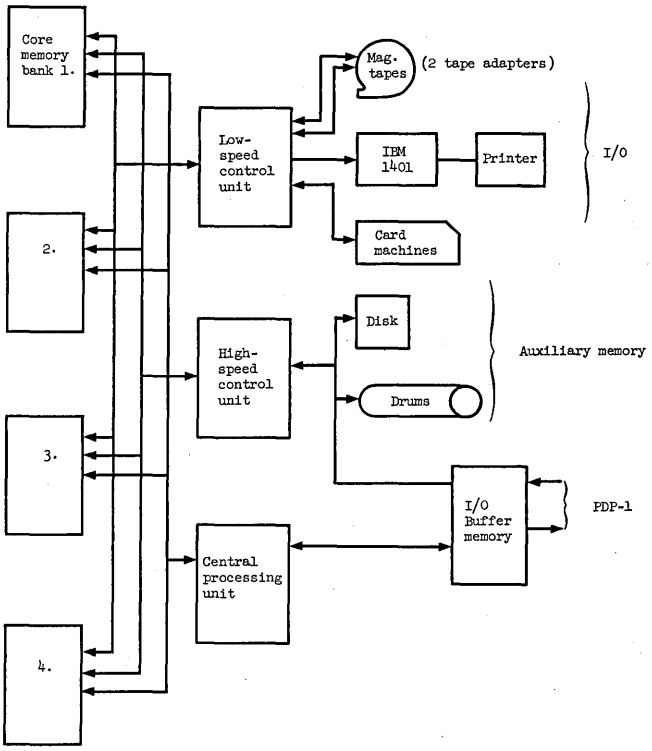


Fig. 1 Characteristics of the AN/FSQ-32 Storage Devices

DEVICE	SIZE	WORD RATE	AVERAGE ACCESS TIME
Core Memory	65K	2.5 usec/wd.	—
Input/Output Core Memory	16K	2.5 usec/wd.	—
Magnetic Drums	400K	2.74 usec/wd.	10 msec
Disc File	4000K	11.75 usec/wd.	225 msec
Magnetic Tapes	16 Drives	128 usec/wd. (High density)	5 to 30 msec (no positioning), depending on whether the tape is at load point, and whether it is being read or written.

the time-sharing system

To understand the services available to a user of the time-sharing system, a review of the system itself would be useful. A simplified description of the system is as follows:

All programs requested by users are stored on an assigned section of the drum until the user quits.

A program is given one "quantum" of time to operate, which represents the maximum time for one operating cycle. The program's turn ends when it requests an input-output transfer, when the quantum of time ends, or when an error condition arises. When the turn ends, the executive system determines if another program is ready to operate. If not, the program will reside in core memory until another program is ready. When another program is ready to operate (and it occupies an area of core which overlaps that of the first), the

executive writes the first back to its place on the drum, and brings in the new program to core.

The process of exchanging programs from core to drum is called *swapping.*

The basic cycle of an execution followed by a swap is fundamental to the time-sharing technique of operation. Controlling this process is a program called the *executive system.*

The executive system resides permanently in 16,000 registers of core. It has the following major functions:

- Responds to all computer interrupts and takes the required action.
- Interprets inputs and commands from teletypes and typewriters.
- Allocates both core and peripheral storage.
- Performs all input-output.
- Schedules the running of object programs (as in the above example).
- Performs a number of on-line debugging services.

Object programs, which can contain up to approximately 47,000 words of storage, are placed in core storage only when they are active (ready to compute). When they are not active (stopped or waiting for input or output), they reside on the drums, placed there by the executive. As shown in Figs. 4 and 5, they are put on drums by the system when given the LOAD command (see below - Basic Commands) via teletype.

The Fix Program, which occupies approximately 2,000 words of permanent storage, reacts to all computer errors, logs the error on a typewriter at the maintenance console, attempts to fix the error, then returns to the executive system with a description of the error condition. The system then isolates the user(s) affected by the error, notifies the affected user(s), concludes the affected program(s), and then continues the normal cycle.

A major function of the executive system is to interpret and respond to the various teletyped commands made by a user. It is essential that most users know these basic commands. They are used to sign in, load, and run object programs, stop programs, and sign out. These commands are:

- LOGIN: The user begins a run. With this command he gives his identification and a "job number."
- LOAD: The user requests a program to be loaded (from either tape or disc). Once this command is executed, the program is an object program to the system.
- GO: The user starts the operation of an object program or restarts the operation of an object program that has been stopped. Once the user gives this command, he can send teletype messages to his object program or the time-sharing system.
- STOP: The user stops the operation of an object program.
- QUIT: The user finishes a particular job. Upon receipt of the command QUIT, the time-sharing system punches accounting information into a card and removes the object program from the system.

Figs. 4 and 5 are examples of this sequence as it would appear on a user's teletype. It will be noted that to each command from a user there is a response on the teletype from the system. Until this response is received, the user cannot assume that the system has reacted to the command. Normally, this response is immediate.

In the brief description of the LOAD command above, it was stated that the requested object program is loaded from tape or from disc. It is assumed, of course, that the program exists in binary form—expected by the executive to perform the LOAD function. There are a number of ways in which a program can be put into this form. Some of the more common will be explained in the section on service routines.

The sequence of actions taken after the LOAD command is given is as follows:

The Executive examines the name of the program, which follows immediately after the command LOAD. If it finds that this program has been stored previously on the disc, it brings it in from the disc, places it on the drum, and responds with $LOAD OK as in Fig. 4. If the program is not on the disc, it types out instructions at the computer console for the computer operators to load this program, or the specified tape reel (optional) for this teletype. It also types $WAIT on the user's teletype. When

Fig. 4. Load From Disc

```
LOGIN 0050 JCX.25
 $OK LOG ON 10
LOAD PRG2            Object program on disc
$LOAD OK             Object program on drum and disc
GO
 $MSG IN.
!STOP
 $MSG IN.
QUIT
 $MSG IN.
```

Fig. 5. Loading a Program From Tape

```
LOGIN 1123 JBX.30
 $OK LOG ON 10
LOAD TPRD 1852
$WAIT.               Object program on tape
$LOAD OK             Object program on drum and disc
GO
 $MSG IN.
!STOP
$MSG IN.
QUIT
 $MSG IN.
```

Fig. 6. User Talking to Object Program

```
LOGIN 0050 JDX.25          User to executive
 $OK LOG ON 10             Executive response
LOAD CALC                  User to executive
$WAIT.                     Executive response
$LOAD OK                   Executive response
GO                         User to executive
 $MSG IN.                  Executive response
 CALC READY                Object program response
+ 2,2                      User to program
 RESULT +4.00000000        Program response
* 3,3,3                    User to program
 RESULT +27.0000000        Program response
!STOP                      User to executive
 $MSG IN.                  Executive response
QUIT                       User to executive
 $MSG IN.                  Executive response
```

this operation is completed by the operator, the program is stored on drum and disc (for subsequent loads), and the response $LOAD OK is typed on the teletype. The time between the $WAIT and $LOAD OK can be a matter of minutes.

"talking" on a teletype

It is quite clear that once the GO command is issued, it is necessary for the user to communicate over the teletype with the object program. Therefore, until otherwise noti-

fied, the executive assumes that any input on the teletype is to the object program, not the executive, and therefore simply passes the input to the program, which reacts in some way, probably with an output on the teletype. Fig. 6 shows a typical sequence of steps taken when a user communicates with an object program.

There usually comes a time when the user must start talking to the executive after he has communicated with his object program—to give the STOP command, for example. The user signals this intent by typing an exclamation point. This action and its consequences are demonstrated in Fig. 6. Correspondingly, it is frequently necessary to recommence talking to the object program, having once used the exclamation point. This is done by typing quotation marks, as shown in Fig. 7.

Fig. 7. User Talking to Both Program and Executive

```
LOGIN 0050 JDX.25
 $OK LOG ON 10
LOAD CALC
$LOAD OK
GO
 $MSG IN.
CALC READY                          Program to user
+2,3,4                              User to program
 RESULT +9.00000000                 Program to user
*4,4
 RESULT +16.0000000
!DIAL 28 HELLO                      User to executive
 $MSG IN.
"*3,3,3                             User to program
 RESULT +27.0000000
!STOP                               User to executive
 $MSG IN.
GO
 $MSG IN.
−4,2                                User to program
 RESULT +2.00000000
!STOP                               User to executive
 $MSG IN.
QUIT                                User to executive
 $MSG IN.
```

other executive services

In addition to responses to basic commands, the executive performs a number of other services for users. Included among these is a set of functions necessary to check out (debug) object programs on-line. The kinds of debugging commands available through the executive include the following:

- *Open*: Displays the contents of the given memory or machine register and uses this as a base address for other debugging commands.
- *Modify Open-Register Address*: Changes the address of the opened register by the given increment or decrement.
- *Insert*: Inserts the given values into the opened register.
- *Mask*: Inserts values by the given mask.
- *Mode*: Displays values according to specified mode (floating, decimal, octal, Hollerith, symbolic machine language).
- *Search*: Finds the register containing a specified value.
- *Breakpoint*: When a specified point in the program is reached, notifies the user, (on options) displays registers, and stops or continues the program. As many as five breakpoints are allowed simultaneously.
- *Dump:* Dumps a given set of registers, either on teletype or tape (to be printed off-line).

The actual commands to perform these functions usually include a symbol or address with one or two unique teletype characters.

Figs. 8 and 9 show examples of on-line debugging operations.

Fig. 8. Examining and Changing Words in Octal and Floating Point

```
40000'/                                  DBUG command
40000' = 0140000000040052'               DBUG response
DATAWORD/                                DBUG command
DATAWORD = +4.56289757E+002              DBUG response
2.5 E-3 *                                DBUG command
$IN                                      DBUG response
```

Fig. 9. Examining and Changing Words in Hollerith and Integer

```
MESGWORD (5) /                           DBUG command
MESGWORD = CONTINUE                      DBUG response
BETA/                                    DBUG command
BETA = −197                              DBUG response
−495 *                                   DBUG command
$IN                                      DBUG response
```

Several commands are available to enable users to communicate with each other. These commands and their functions include:

- DIAL: Permits sending messages to any other user in the system.
- LINK: Initiates linkage of any two teletypes so that they act as one. Both teletypes can input and output to and from the same program or the systems. Also, both teletypes can type all the information being input or output on each other.

Commands to query the system about its status are available through the executive. Examples of these are:

- USERS: The response to this command is the number of currently active users in the system.
- TAPES: The response to this command is the number of tapes available for use by object programs.
- DRUMS: The response to this command is the amount of drum space available for object programs.

Quite clearly, object programs must have access to the on-line input-output devices that are part of the system. In addition to these, the system at SDC permits access to disc files and tapes as completely as is possible with the computer. Since a "traffic" problem exists with these devices (as well as with the on-line devices), all input-output units are assigned by the system. No object program makes an absolute (machine) reference to an input-output unit. In the object program, requests for input-output "files" include arbitrary names, for which assignment of a particular unit is required. The executive then assigns an actual (machine) unit or an area of a unit such as disc (if one is available). For subsequent reads, writes, or positionings of the unit, the object program requests the executive to move the file by using the name given in the initial request. Any attempt by an object program (in machine language) to read or write a unit directly will result in a computer interrupt that will stop the program.

A mechanism has been provided to permit the multiple use of one program by several users. An object program may in its input-output file declarations request more than one teletype or display console (in this case, specific units). Thus a single program can act as a recorder, monitor, situation generator, or play some other central role to a group of users in a game or other team effort. *(To be continued).* ■