THE SDC TIME-SHARING SYSTEM

part two: service routines & applications

by JULES I. SCHWARTZ

With the commands and functions discussed so far, a user can load, run, and debug programs, as well as have other miscellaneous services performed. It is, of course, necessary that numerous other services also be provided. The functions described so far were performed entirely by the executive system. The object programs were considered to be just code running in response to basic commands. However, the techniques for providing additional services are actually accomplished through the use of object programs. Object programs can be written to provide other necessary services; such programs are called service routines. Since service routines are object programs—and there is no limit to the number or kind of object programs that can be used—there is in effect no limit to the number of services that can be provided.

Service Routines for Producing Object Programs. In some respects, the most common technique for producing object programs is similar to that used in standard computing installations. The program is written in a symbolic language, stored on a magnetic tape (or disc), and then compiled. The output of the compilation is a binary program—in this case compatible with the executive LOAD command—and a listing. When the symbolic programs require modification, the necessary changes, deletions, and insertions are made by using the tape (or disc file) that contains the program, and a new tape (or disc file) is prepared.

File Maintenance

The preparation and maintenance of symbolic tapes and disc files is done with the service routine called FILE. The functions of the routine FILE are:

- *Generate Symbolic Files.* Symbolic files may be stored on tape or disc from teletype inputs or through the card-reader.
- *Update Symbolic Tape Files.* Symbolic files may be updated on-line without destroying the original file. Using the update feature, lines may be inserted, deleted, or replaced via the teletype.
- Merge Symbolic Tape Files. An additional feature of the tape update portion of FILE allows files from a second tape to be merged with files of a first, base tape. The file to be merged may be inserted at any point within a file on the first tape.
- *Print Symbolic Tape Files.* Symbolic files or parts thereof may be listed either on the user's teletype or on an output tape for later printing off-line. This feature may also be used to make extracts or duplicates of symbolic tapes.
- *Survey Symbolic Tapes.* To review the contents of a symbolic tape that contains a number of files, the user may wish to survey the tape. A request for this operation will cause FILE to search the tape and to print the first "*n*" lines of each file on the user's teletype.

Compilers

There are several compilers available in this system. The JTS compiler was designed to provide JOVIAL and SCAMP (machine language) compilations under the time-sharing system. JTS accommodates a subset of the JOVIAL J-2 and J-3 languages as well as a subset of SCAMP. The compiling function of JTS can be performed on-line, in a sense, if the user wishes to wait at his teletype and review any coding errors that JTS outputs on the teletype. In addition, the user can operate his object program immediately after successful compilation. The binary object program produced by JTS is on a tape that conforms to the format requirements for system loading. The user can specify the type of program listing to be output on his listing tape.

A second compiler available to users is called SCMP. It has the same operating characteristics as JTS; however, the language it compiles is the complete SCAMP.

Other compilers, including SLIP and LISP, are either available or are being implemented.

IPL-TS

A somewhat different scheme is provided by the service routine IPL-TS. In this case, the object program (coded in IPL-V), prepared through use of FILE, is assembled by this service routine. The assembled program is then made part of the IPL-TS system, which can be saved (on option) and later reloaded with the LOAD command. When it has been loaded, the program may be executed interpretively by the IPL-TS service routine. This routine also provides a great number of on-line checkout aids to the user during execution of his program.

The service routines and techniques discussed so far permit users to produce and modify both small and large programs in a manner analogous to other kinds of computer systems although they are generally controlled on-line. Techniques more appropriate to time-sharing systems are also available for producing, checking out, and running programs. These techniques provide the capability for coding and executing programs on-line (at the teletype) without going through the various independent steps necessary in the file and compile process. The service routines now available for this purpose are called TINT and LIPL.

With TINT, one may program in the JOVIAL language; LIPL provides the ability to program in the IPL-V language. In both routines, execution is done interpretively, providing many on-line debugging and communication aids that are not available when executing a binary program in the normal fashion. The general description of these programs is as follows:

TINT was developed to provide a vehicle for on-line coding and execution of JOVIAL programs. The applications of the on-line interpreter are:

- Program composition
- Debugging and editing
- Rapid formulation and computation

Functions

- To accept, perform legality (grammar) checks on,

and interpret statements to a given subset of the JOVIAL language.
- To permit execution of all or part of small JOVIAL programs.
- To permit dynamic input of variables to a JOVIAL program that is to be executed.
- To permit dynamic output of results obtained through execution of a program.
- To permit on-line symbolic corrections to be made to existing code.
- To permit storage of symbolic code composed with TINT and then transferred to tape so that it may later be compiled or re-executed interpretively.

Fig. 10 shows an example of a small TINT program as coded and executed on-line.

The IPL-TS interpreter described earlier also permits the programming and execution of on-line coded programs. In this case, the technique is analogous to that used in programming off-line (using tape input) except that the code is assembled from teletype input. Also, with this routine, programs prepared with the FILE program and assembled from tape can be modified by program input on a teletype. A brief example of a LIPL program is given in Fig. 11.

Fig. 10. Example of a TINT Program

```
$  1 "THE EUCLIDEAN ALGORITHM"
$  2 "GIVEN TWO POSITIVE INTEGERS A AND B"
$  3 "FIND THE GREATEST COMMON DIVISOR"
$  4 S1. READ A,B;
$  5 X = A; Y =B;
$  6 S2. IF X EQ Y;
$  7 BEGIN PRINT 30H(THE GREATEST COMMON
       DIVISOR OF),
$  8 A,3H(AND),B,2H(IS),X;
$  9 GOTO S1; END
$ 10 IF X LS Y;
$ 11 BEGIN Y = Y-X; GOTO S2; END
$ 12         X = X-Y; GOTO S2;
$PRINT COMPLETE
$ENTER COMMAND
?EX
     A = ? 1024
     B = ? 512
THE GREATEST COMMON DIVISOR OF 1024 AND
  512 IS 512
     A = ? 234
     B = ? 86
THE GREATEST COMMON DIVISOR OF 234 AND
  86 IS 2
     A = ? 234
     B = ? 84
THE GREATEST COMMON DIVISOR OF 234 AND
  84 IS 6
     A = ? 234
     B = ? 82
THE GREATEST COMMON DIVISOR OF 234 AND
  82 IS 2
     A = ? 234
     B = ? 80
THE GREATEST COMMON DIVISOR OF 234 AND
  80 IS 2
     A = ? 234
     B = ? 78
THE GREATEST COMMON DIVISOR OF 234 AND
  78 IS 78
```

A number of other routines exist and are being written for use in the SDC time-sharing system. These include routines whose functions range from information retrieval to tape copying.

applications

Thus far this article has described the characteristics and capabilities of the time-sharing system in use in the Command Research Laboratory. The system has been in operation since June 1963, after an initial development of approximately five months. Currently it is operating eight hours a day and is virtually the only means for using the computer during the day in the Command Research Laboratory.

Fig. 11. Example of a LIPL Program

```
LIPL-READY

RT SO = (AO 10MO 109-3 J100) 9-3 = (40HO J75 J71,J151)
AO = (11MO J50 10MO J68 10PO 709-5,9-1 J60 709-6 12HO 61MO
51MO,9-2 J60 709-3 12HO 52HO 12MO J2 709-2 30HO,9-1)
9-3 = (30HO AO 70 (108.0 21MO,9-13,9-4)
9-4 = (40MO 51WO 20MO,J30) 9-5 = (J4,9-4) 9-6 = (108.0 21MO,9-4)
DT PO = (0 $RED$ $WHITE$ $BLUE$ $GREEN$) MO = (0 C1 C2 C3 C4 C5 C6)
C4 = (0 C1 C3 C5 C6) C5 = (0
C1 C4 C6)
C6 = (0 C1 C2 C3 C4 C5) NL GT SO
```

C1	0232254	21	RED
C2	0236814	21	WHITE
C3	0236869	21	BLUE
C4	0236814	21	WHITE
C5	0236869	21	BLUE
C6	0236848	21	GREEN

An IPL-V "map-coloring" program, written and executed on-line under time-sharing in Linear IPL (LIPL).* This program determines the colors of all countries (symbols C1 to C6) on a map (list MO) such that no adjacent countries are colored alike. The six-country map for this example is configured as follows:

C1 C2 C3 C4 C5 C6

*LIPL was written by Robert Dupchak of the RAND Corporation.

Of some interest are the numerous and diverse applications of the system. These serve to show the possibilities offered by the present and relatively young system as well as to point out the large range of applications and services that a powerful concept such as time-sharing can provide. A list of some of the current applications in the Command Research Laboratory follows:

- Natural Language Processors—used for parsing English sentences, answering questions, and interpreting sentence-structured commands.
- Group Interaction Studies—in which teams of players are matched against each other, and in which the computer is used to measure individual and team performance.
- General Display Programming—in which the programs are used as vehicles for generating and modifying visual displays according to the user's keyboard inputs.
- Simulated Command Post—a realistic simulation of a command post has been produced, and such problems as the display requirements for this organization are studied within this framework.
- Hospital Control—the data for a ward of hospital patients is maintained and retrieved through the system, with access from stations in the hospital.
- Text-Manipulation—a sophisticated text-manipulation program has been developed.
- Police Department Crime Analysis—using some of the techniques found in the studies of natural-language

comprehension, reports of crimes are compared with a complete history of criminal reports to establish patterns and isolate suspects.

- Personnel File Maintenance—personnel records of SDC are maintained and accessed during the time-sharing period.

comments and prospects

The results of the first year's use of the SDC time-sharing system have been encouraging. A considerable amount of work has been accomplished using it, a great deal has been learned about the problems of time-sharing, and a number of applications have had a great deal of exercise which could not have been attempted with more traditional computing center techniques.

The actual development of the system has been in roughly four stages, the first three of which lasted about six months each. These stages are:

- Design and Checkout of the Initial System—During this period the emphasis was on the executive system, with only a slight effort in designing service routines.
- Initial Use of the System—The main concern during this period was making the system "stay alive." (It frequently didn't, causing numerous frustrations and feelings of ill will toward time-sharing). The majority of users during this period were members of the Time-Sharing Project who were writing and checking out service routines. A few of the applications systems were begun during this period. The number of services and conveniences for the user were minimal. The system was in operation between two and four hours a day.
- Full-Scale Use of the System—During this period, the time-sharing period operated eight hours a day. A large number of applications were programmed, checked out, and used. The set of service routines was expanded, and the ones that existed were sharpened considerably. A number of the "little annoyances" of the system were eliminated, and in general the system was made much more reliable and easier to use. During these three periods, a large number of changes to the equipment was made. Probably a significant change was made on the average of once every six weeks. This, of course, did not aid the reliability of software or hardware.
- The Future—We are currently in the fourth phase of this system. Changes to hardware should be relatively few now, so that the software emphasis can be on improvements; there are seemingly an infinite number of improvements possible. They range from such ideas as telling the user his program's status and the time of day when he asks for them to improved executive input-output buffering schemes and techniques which permit a user instantaneous access to a network of programs on other computers as well as the Q-32. (There is currently a list of over 50 such items waiting for implementation).

The fact tnat so much remains to be done might lead one to the conclusion that the concept has not been very satisfactory. On the contrary, we can probably say that our experience so far with time-sharing has proven quite satisfactory, and the true potentials of such a system are now becoming clear and realizable.

When "discussions" of time-sharing (and on-line computer usage) are conducted there is generally agreement on the use of the concept for a number of applications, but there is considerable debate concerning the "economics" of it—whether more traditional computer systems make more efficient use of the computer. Like many such questions, the answers cannot be found easily. Time-sharing permits many runs on a computer and instantaneous response to all users. It also encourages techniques which are quite valuable but not practical otherwise (e.g., solutions by trial and error, use of displays, on-line debugging, single-shot retrieval of information, etc.). In some respects, it makes excellent use of a computer. For example, since there is almost always "something" going on, time to mount and demount tapes is never wasted time.

On the other hand, it can be pointed out that in the "worst case" of time-sharing today—where big programs must be swapped frequently—the efficiency of time-sharing is low. (This applies primarily to efficiency of throughput, not response time, which is another measure of time-sharing efficiency). For certain kinds of programs—those which require long periods of compute time and where human interaction cannot help the process—time-sharing is of no direct value.* Time-sharing and on-line computer use tends to discourage or make difficult retrieval of large quantities of printed output. Although time-sharing assists man-machine interaction by letting users use the computer on-line, it also frequently requires humans to be present at jobs they would be quite happy to let run without them.

In the system at SDC, certain of these arguments are recognized. However, at the present time, they do not represent serious difficulties. The throughput and response time for the system are quite adequate for a reasonably heavy load. If the capacity of the system were to be increased (primarily by increasing the size of the drums), there seems little question that, without considerable improvements in the system, the economic factors would be more serious. Thus, although we have been able to tolerate a close to "worst case" scheduling mechanism in the early phases, areas of unoverlapped swap and input-output will have to be eliminated with a larger average load. Also, the running of programs in a "background" fashion, so that humans aren't required and long computations don't unnecessarily degrade the system, is an item of high priority in the future.

In conclusion, one can view the present system and the experience so far and have a great feeling of optimism for the future. Emphasis from now on will be in areas that will stress significant improvement in the techniques and tools available to the user. The problems of hardware modification, hardware and software reliability, and others due to lack of experience or haste in production are diminishing. Even with these various areas of growing pains, a surprising amount has been accomplished.

Time-sharing seems to hold a key to much that has been bothering the computer using community. The computer can be brought close to the user. Problems not heretofore solvable can be pursued. The problems of economy in some areas are better now with time-sharing, and in others no impossible problems seem to exist. Large-scale use of computers on-line seems to be with us to stay.

BIBLIOGRAPHY

This Bibliography contains additional information about the System Development Corporation Time-Sharing System.

1. Rosenberg, A. M. (ed.) *Command Research Laboratory User's Guide.* SDC TM-1354 Series, November 1963.
2. Schwartz, J. I., E. G. Coffman, Jr., and C. Weissman. *A General-Purpose Time-Sharing System.* SDC SP-1499, 29 April 1964.
3. Schwartz, J. I., E. G. Coffman, Jr., and C. Weissman. *Potentials of a Large-Scale Time-Sharing System.* To be published in the Proceedings of the Second Congress of Information System Sciences, November 1964. (Also available as SDC SP-1723.)

*There is the possibility that the compute time can be cheaper when shared than when alone.