BESYS revisited

by R. E. DRUMMOND
AT&T Bell Laboratories
Holmdel, New Jersey

ABSTRACT

The origins and development of the BESYS family of operating systems are traced. Developed at AT&T Bell Laboratories in the late fifties, the system was used for over ten years to control and facilitate the use of the IBM 704-709X series of computers. Some of the novel operating system techniques created for the system and the people who produced them are chronicled.

INTRODUCTION

In the operating system sphere, Bell Labs is best known for UNIX.™* But before there was a UNIX operating system, there was an operating system called BESYS.** For over ten years BESYS was a mainstay for computing at the Labs and altogether served our users from late 1957 to early 1971.

The UNIX operating system has merited much attention in the literature. In contrast, BESYS is all but forgotten. A search of the available literature turned up a single one paragraph entry in *Encyclopedia of Computer Science and Technology* that mentioned BESYS.[1] Even the internal Bell Labs literature is scant when it comes to recording BESYS contributions. Once-common documents that described the system and how to use it have all but disappeared, having somehow escaped the document archives.

This paper rectifies that oversight and attempts to set the record straight about the development of BESYS.

THE PRE-BESYS PERIOD

Bell Labs has been involved with computers since the early 1930s. The computing establishment that gave rise to BESYS, however, consisted of a half dozen members of Bell Labs' Mathematics Research Center. In 1952, they acquired their first commercial computer: a small IBM Card Programmed Calculator.[2,3] This group included V. M. Wolontis and R. W. Hamming who, when IBM 650s were installed starting in 1955, developed the once popular problem oriented languages, L1 and L2.[4,5]

L1 and L2 extended the 650's capabilities by interpretively providing floating-point arithmetic, simplifying memory access, and supplying useful diagnostic information; thus enabling users to straightforwardly program the underlying machine. This encouraged users to program their own applications, allowing them to realize the right results faster and to benefit from the feedback provided by the programming experience. The essence of this "open shop" approach to computing was captured in a well known Hamming motto: "The purpose of computing is insight, not numbers."

The operation of the equipment itself, however, was "closed shop." Since 650 operations were not complex, users could set up their jobs to be run by professional operators. This permitted the machines to be run efficiently on an around-the-clock basis.

By mid-1957, to handle the growing computing workload and increasing complexity of applications, an IBM 704 was ordered to replace the 650s, and preparations for the new machine began that were in keeping with the philosophies that had evolved.

EARLY 704 ERA

The 704 had been announced in mid-1954, and the first one was in the field by the end of 1955.[6] So by mid-1957, many programming tools existed to facilitate its use. By then the SHARE 704 users group was two years old and was already an important source of programs developed by its members. Through SHARE, users built programming tool kits, I/O, and mathematics libraries enabling them to write, debug, and execute their own application programs using stable facilities. In addition, FORTRAN was about to make its debut.

Some of the more useful programs available through SHARE are shown in Table I. At that time, programs were written in SAP or FORTRAN. They were translated to binary machine level usually as card decks. These decks could be combined with those of previously translated I/O or mathematics routines and loaded into the computer using a tool like NY-RBL.

Execution, sometimes with a debugging tool like NY-SNAP, produced print or punch output. For large amounts of input or output, magnetic tapes usually came into play employing off-line tape drives, printers, card readers, and punches where appropriate.

Characteristically, most tools were oriented to stand-alone use and only worked together through manual intervention by the user or a trained operator. Few operating systems existed or were in use at that time.

A definite mismatch was widely recognized between the 704's internal speed, the sluggishness of its on-line unit-record equipment, and the inherent slowness of manual operations associated with stand-alone use. Clearly a new mode of operation was required for the 704 if it was to be used in an effective and efficient way.

EMERGENCE OF BESYS

Soon after Bell Labs placed its first 704 order, G. H. Mealy and Gwen J. Hansen addressed the problem of using the machine by developing one of the earliest operating systems. Their solution was a system that provided:

*UNIX is a trademark of AT&T.

**The name BESYS has reportedly stood for BEll SYStem which although appropriate, is perhaps too grand. It actually is derived from a running together of BE and SYS. BE was the SHARE assigned installation code for Bell Telephone Laboratories, Murray Hill, NJ and SYS stood for system, which was indicative of the programming classification for BESYS. This was consistent with SHARE naming conventions for programs it distributes.

1. Flexible operator control of the hardware normally in a non-stop mode of operation
2. Efficient batch job processing usually in a tape-to-tape operating mode through use of off-line unit-record equipment
3. Effective user access to system facilities using control cards to minimize user-operator interaction
4. User program access to centralized I/O, system control, and elementary mathematical functions

TABLE I—Some SHARE programming tools. The installation codes shown in this table served to identify the tools' contributors. Thus, UA stood for United Aircraft; PK, IBM Poughkeepsie; NY, IBM New York; and so on.

SHARE Programming Tools		
Installation Code	Program ID	Functional Description
UA	SAP	Symbolic Assembly Program
PK	CSB 4	Card to Storage Binary (Loader)
NY	RBL 1	Relocatable Binary Loader
UA	CSH 2	Card to Storage Hollerith
UA	SPH 3	Storage to Printer Hollerith
UA	DBC 1	Decimal to Binary Conversion
UA	BDC 1	Binary to Decimal Conversion
NY	SNAP	SNAPshot dumper
NY	SNAQ	Snapshot Output Converter
UA	TAD 1	Tape And Drum utility
UA	TPH 2	Tape to Printer Hollerith
CE	650W	650 Wire Plugboard Simulator
CE	650S	650 Simulator
UA	STE 1	System Tape Editor
UA	LST 2	List Tape Converter

5. Snapshot and post-mortem dump debugging facilities
6. Compatibility with the 650s by simulating the L1/L2 interpreters

The initial system, BESYS-1, was in use by October 16, 1957.[7] It was designed for a 704 with drums, 8K 36-bit words of core memory, a full complement of tapes, and on-line/off-line unit-record equipment. The system was developed using facilities at the Esso Research Center. Ironically, it would never be used at Bell Labs because our original order changed before delivery to a drumless 704 with 32K words of memory. This change gave rise to BESYS-2.[8,9]

Both systems were essentially identical. They usurped the lowest 64 and highest 4K words of memory, a subset of the tape drives and the sense switches. In addition, BESYS-1 swapped a portion of the upper 4K memory to drum storage making it available for user program variable storage. The lowest 64 locations were reserved as a hardware communication area; the upper 4K locations were used for the core resident portion of the system. The complete system resided on tape and consisted of FORTRAN and most of the tools in Table I adapted to work in the BESYS environment as subsystems or integrated into the core resident portion of the system itself.

One specific aim of these changes was the removal of program stops when error conditions were detected. The stops were replaced by messages describing the condition and a standard recovery taken when possible. If operator action was required, the messages appeared on the on-line printer and a standard system stop occurred. This helped the operator distinguish random program stops from planned stops and contributed to the non-stop mode of operation.

BESYS OPERATIONS

Although much effort was devoted to reducing the involvement of operators in the processing of jobs, the operator still played a key role in the use of the machine. At the meta-system level, the operator was responsible for:

1. Starting the system
2. Batching jobs for system input
3. Assigning tape drives and mounting tapes
4. Responding to system error messages
5. Batching output for peripheral processing
6. Terminating jobs that stopped, looped, or exceeded job limits
7. Recovering from system failures or corruption

The heart of the system was the core resident portion. It contained the following:

1. System control program
2. Centralized I/O facilities
3. Comprehensive mathematical functions
4. Tables and buffers

The system control program functioned at several levels: (1) operator, (2) user, (3) program, and (4) hardware.

Operator Level

At the operator level, BESYS used the sense switches and the on-line printer as an interface to process a sequence of jobs. Jobs were presented to the system by the operator a "batch" at a time through the standard input stream. Normally, the medium was tape, and the system would sequence through the jobs in rapid succession. Each job was logged on the printer as it was encountered, permitting the operator to monitor the system's progress. The system also logged detected error conditions on the printer, sometimes soliciting an operator response. The printer could also be used by the operator to monitor the standard output stream. The operator used the sense switches to:

1. Select the standard input source: tape or on-line card reader
2. Alter normal sequencing of jobs in the standard input
3. Provide a go/no-go response to error conditions
4. Monitor output

User Level

At the user level, BESYS implemented a process that read user-provided control cards from the standard input and interpreted their contents to determine the system functions to exercise. The control cards looked just like SAP statements and consisted of an optional location field followed by an operation field and possibly a variable number of operands. The operation field specified the system function. The location and operand fields were passed to that function for further interpretation.

On initial entry, the control card processor searched the standard input for a JOB card. Once found, the system initialized itself to process the incoming job. It then read the next control card and performed the specified function. These functions included:

1. Invoking sub-systems like SAP and FORTRAN
2. Loading object programs
3. Patching loaded programs
4. Accumulating snapshot dump requests
5. Communicating to the operator via the printer

Control card processing would continue until a TRA card was encountered. The TRA function would complete any unfinished business associated with program loading, plant any accumulated snapshot dump requests and TRAnsfer control of the machine to the user's program.

Following is an example of a job that would have caused BESYS to do a FORTRAN compilation with test.

```
      JOB account-number, programmer-name
      FOR
(FORTRAN source program statements)
      LOD 4
      TRA
(Data cards for program test)
```

The JOB card signified the start of the job and identified the user in terms of an account number and programmer name. The FOR card invoked the FORTRAN compiler which would have processed the following source program leaving the resulting object program on tape 4. The LOD card loaded the object program from tape 4 into memory. Finally, the TRA card started the loaded program's execution.

Program Level

At the program level, BESYS provided many services. Primarily, they consisted of: I/O routines, snapshot dump routines, system control routines, and, initially, the common elementary mathematics routines. Nearly all the facilities normally available to the FORTRAN programmer were present in the core-resident portion of the system. The rationale for this was that the SAP programmer could also make use of them, and, since most programs would use them, the memory they occupied was not wasted. Also, the time saved by not having to load them repeatedly, contributed to over-all system efficiency.

Eventually, introduction of FORTRAN II and development of a program linking capability for the loader weakened this argument, making it reasonable to purge the mathematics routines from the resident portion of the system. This also proved to be a necessity because, as BESYS matured, free space within the system's 4K region became a critical resource.

Once control transferred to the user's program, overall control became tenuous. There was no hardware facility that enabled the system to constrain the user's program; their relation to one another was peer-to-peer. Hence, the system was at the user's mercy. Only honor and social pressure bound the user to conventions established by the system, and only the operator's sensitivity to how the system was behaving remained as the effective means to limit damage when things went awry.

Although the peer-to-peer relationship represented a potentially critical integrity exposure, in practice it did not prove to be a devastating factor. Because jobs were processed one at a time, the user that violated the system usually stood to suffer the most. Nevertheless, it was a weakness that forced a greater dependence on the operator than desired.

At least two facilities were provided for programs to return control to the system. The first, RETURN, simply returned control to the control card processor initiating a search for the next valid control card and resumption of control card processing. This facility made multi-step jobs possible. The second, ENDJOB, initiated end-of-job processing. Any post-mortem dump requests were taken, the snapshot converter was invoked, miscellaneous end-of-job housekeeping performed, and, finally, the control card processor was reentered to search for the next valid JOB card.

Hardware Level

At the hardware level, BESYS had to deal only with floating-point traps. Facilities were provided that examined the floating-point "spills" and either produced appropriate

diagnostic messages or honored alternative actions specified by the user through use of ALT control cards.

SYSTEM ACCOUNTING

The 704 had neither a time of day clock nor a timer to simulate one; hence, usage accounting was crude. A time card accompanied each job submitted to the system. It served to:

1. Identify the user who submitted the job
2. Provide job run time limits and output estimates
3. Specify special handling requirements such as tape IDs and their logical tape drive assignment or paper and card stock to use for peripheral equipment

The card or the information it contained was also used by operations to: (1) schedule jobs, (2) fetch and set up tapes, (3) set up peripheral equipment, (4) feedback written remarks from operators, and (5) log time on and off the system using a manual time-clock.

Eventually, usage accounting was improved through use of an electro-mechanical time-of-day clock tied to the on-line printer. Accounting information was then punched on-line into cards as part of the ENDJOB housekeeping function.

BESYS DISTRIBUTION

True to its origins, BESYS 1 and 2 were submitted to SHARE for distribution and were also made available to several external installations directly. Reportedly BESYS was used by many installations,[10] but no records remain on which ones they were. Only faint recollections suggest they included The Esso Research Center, Mobile Oil Corporation, and The Shell Oil Company.

THE BESYS-2 SUCCESSORS

G. H. Mealy left Bell Labs for the Rand Corporation in 1959 just as plans were being formulated to develop BESYS-3, the first of a series of successors to BESYS-2. G. J. Hansen and the author continued this effort.

BESYS-3

BESYS-3 was a port of BESYS-2 to the IBM 7090. The major efforts involved revamping system I/O facilities, replacing SAP with FAP and upgrading FORTRAN from the 704 to the 709X version. It was placed into service in July, 1960.

In the process of porting, BESYS received a general face lifting. New control functions were added; others were dropped because their function had become obsolete or was done in some other way. Some control cards felt the effects of human engineering. The LOD control card for invoking the loader, for example, became the LOAD control card. The so-called Yum-Yum cards*** of NY-SNAP vintage were renamed and reformatted and came out as SNAP and COREPM.

Upgrading FORTRAN was no trivial task but in general it was straightforward. Replacing SAP by FAP, on the other hand, was more complex. By 1959, what started out as UA-SAP under BESYS had been significantly extended, and although FAP shared the same ancestry, it lacked these extensions. The most important extension was a conditional and recursive macro facility developed by D. E. Eastwood and M. D. McIlroy.[11]

Since FAP addressed the new facilities provided by the 709X hardware and was already in use on that hardware, our SAP extensions were ported to FAP; in effect, merging the best of both assemblers. The result became widely known as BE-Macro-FAP and eventually became the basis for IBM's IBSYS assembler: MAP.

Although the 704 BESYS environment encouraged using centralized I/O facilities, the peer-to-peer relationship of system and user programs enabled sophisticated users to start their own I/O directly and exploit the CPU-centered I/O copy-logic to process data on the fly as it was transmitted to tape, the CPU, and core. The architecture of the 709X machines, however, separated the data transmission function from the CPU by using independent channels. This provided the potential for a high level of concurrency if programmed properly. In addition, the channels could signal the occurrence of major I/O events by interrupting the normal sequential processing of the CPU.

Fully realizing the concurrency potential required the development of a sophisticated set of interrupt driven I/O routines. Even though only a simple buffering strategy was employed initially using such an approach,**** the implementation technique virtually precluded coexistence with user written I/O routines. Thus, users were forced to use the system's centralized I/O facilities.

This had two effects: (1) the user level I/O facilities had to be beefed up and (2) the amount of memory required by the resident system increased sharply. This led to the controversial decision to double the system area of memory from 4K to 8K, even though the maximum memory of 32K remained the same for the 709X machines.

Some arguments proposed, not unreasonably, that the 4K limit on system area size be preserved by purging from the resident system all the higher level FORTRAN based I/O

***So called because the handwritten words "Yum-Yum Cards" were scrawled across the top of a SHARE program write-up of NY-SNAP, presumably by G. J. Mealy, that was circulated within Bell Labs during the early 704 days. All the NY-SNAP cards were of the form Y- - where '—' stood for some condition imposed on the dump request. For instance, YUN—dump UNconditionally, YPL—dump if the accumulator was PLus, YPM—provide a Post-Mortem dump, etc. They were presumably good to feed to the system when debugging, whence Yum-Yum!

****Apparently BESYS-3 was an early user of interrupt driven I/O. This was based on our testing experience at several different 709 sites. Each time we tested at a new site we had to have jumpers changed in the Data Synchronizers to activate their Data Channel Trap feature.

*****Except for the type information, the blocking scheme was equivalent to what is currently called "Variable Block Span Format" in IBM's OS/370.

routines just as the elementary mathematics routines had been purged under BESYS-2. From a public relations point of view this would have looked good for the system, but from the user's perspective it was really a trade-off—user space for system space. Thus, although it would have been a challenge to live within the original 4K size, as an expedient the additional memory was taken for system use. Time would show that as the system continued to develop there would be no problem in using all that additional space.

A new and important dimension to computer output was introduced with BESYS-3 when C. F. Pease developed a basic software package to generate tapes that could drive a Stromberg-Carlson 4020 microfilm printer. This device was capable of producing high quality graphics. It enabled users to visualize their data, providing them with new insight. Soon after its introduction, it was even used to produce computer-generated movies.

One continuing goal of BESYS was to improve the quality of feedback to users about their programs. Along these lines, V. A. Vyssotsky in 1961 conceived and implemented a preprocessor to FORTRAN that analyzed source programs looking for use of variables before their initialization and sections of code that could not be reached. Typically, these problems went undetected by the compiler, yet frequently produced subtle and sometimes mysterious results at execution time. This improvement to FORTRAN was welcomed by our users but was treated with indifference when offered to IBM.

BESYS-4

Once the conversion to BESYS-3 was completed, thoughts turned to how the system could be made more efficient, how the operator interface could be improved, and how the system could be made more robust.[12]

Buffered and Blocked I/O

System efficiency was given a boost by moving from the simple buffered I/O strategy to one that was completely general, permitting read-aheads and write-behinds to an arbitrary depth. There was also a move away from unit record processing to a generalized blocking scheme employing logical records of arbitrary length and containing indications about the type of information being handled.***** Blocking effectively increased tape capacity and significantly improved data transfer rates.

The blocking format was designed with IBM's 1401 in mind so that the 1401s could process tapes used for the standard I/O streams. The logical record type enabled both print and punched output to be merged into the same stream along with job accounting information reducing the number of dedicated on-line tapes and eliminating the need for an on-line punch. User data tapes could also be dumped when necessary using the 1401s.

Unless the user took some deliberate action, all I/O was automatically blocked and buffered in a transparent fashion. Even the unused area of user programming space was automatically used for buffers. From the user's perspective, all I/O was done on a logical record basis without regard to detailed questions of hardware efficiency; that was deemed to be the system's job.

Hardware Extensions

To attack some of the major problems of operating the equipment and controlling the flow of work to and from the system as well as provide a high-speed data connection to the system, G. L. Baldwin and H. S. McDonald collaborated to build equipment that interfaced to the 709X systems using the Direct Data Feature. This equipment provided the following facilities:

1. Interval timer
2. Time-of-day clock
3. Day-of-year calendar
4. Operator's display panel
5. Tape control system
6. High speed data link

Interval timer

The interval timer eliminated the need for the operator to monitor the running time of jobs streaming through the system. The user now entered the estimated maximum run time to the nearest one hundredth of an hour on the JOB card and the operating system automatically cut the job if it was exceeded. A visual analog display of the amount of execution time remaining for the job on the system was provided to assist the operator with scheduling.

Clock and calendar

The clock and calendar were useful in providing accurate information for accounting and other time stamping applications. The clock provided user programs with the ability to measure time with full millisecond accuracy.

Operator's display panel

The operator's display panel consisted of five system status indicators, two with integrated push buttons and a programmable speaker. Four of the indicators reflected major system states: FAP, FOR, RUN, and SYS. The fifth indicator reflected the HOLD/RELEASE state of the interval timer. The RUN indicator was also a button. When lit, depressing it reset the interval timer and thereby normally terminated the job in progress. Using the button associated with the HOLD/RELEASE indicator, the operator could exercise a manual override of the interval timer. The speaker was driven by a flip-flop and was normally used by the system to produce an audible tone alerting the operator when some condition required manual intervention.

Tape control system

The tape control system provided BESYS with almost the same capabilities as the operator to control tape drives within the computing center. Each tape drive was provided with a control panel that displayed the unit's status and which the operator used to enter reel numbers, select unit addresses, switch drives on- or off-line, and set special status information. Except for entering reel numbers, the operator could also perform the same operations from a centralized console. Any change made by the operator using these facilities caused a manual entry latch to be set. This, in turn, caused the system to update its internal tape-drive-state table.

The operator still had to mount (taking into account a user specified channel), identify, and unmount tapes for the system, but from that point on the system did the rest, namely:

1. Read tape reel numbers entered by the operator
2. Switch the drives on- or off-line
3. Assign or sense unit addresses on drives
4. Ready or unready drives
5. Sense status indicators set by the operator

Drives mounted with private tapes were always placed in a reserve state. This distinguished them from drives that could be used for system or scratch purposes. The reserve state was also used with batched output tapes as an operator request for the system to close out the tape and queue it for peripheral processing. The next tape would then be automatically selected, switched on-line, and readied for use by subsequent jobs—all in a matter of seconds. Similarly, at the end of a batched input tape, the system would switch to a new one. The selection criteria were based on a unit address convention followed by the operator. The operator would be prompted only when a suitable successor couldn't be found.

The user conveyed non-system tape requirements for a job using MYTAPE control cards. For a given logical unit, they specified the tapes to use by reel number (multi-reel tapes were accommodated) and whether the tapes would be used for input, output, or scratch. Input tapes were file-protected and marked read-only to the system I/O routines. The tape control system proved to be an effective tool and significantly improved tape handling and the flow of work.

High-speed data link

The high-speed data link connected BESYS to a Packard Bell 250 computer located in a remote laboratory. Users within that laboratory could request the interjection of a job into the current job stream to process data transmitted through the data link.

Other monitor aides

Several RPQs were added to the 709X hardware to improve system robustness. One was a trap on halt instructions. Another was a crude form of memory protection that was used to protect BESYS from vagrant stores and, under the right conditions, was also used to protect buffer pools set up in user areas. Both improvements enhanced the system's control and reduced the monitoring role of the operator but did not alter the peer-to-peer relationship enjoyed by the knowledgeable user.

BESYS-4 went into operation in April, 1962.

BESYS-5

Like most of its predecessors, the production of BESYS-5 was motivated by hardware changes. On the surface, support was introduced for the CPU extensions associated with the IBM 7094 and for the IBM 1301 Disk Storage Facility. Internally, however, the changes called for major revisions of the system's I/O routines to cope with the 1301s and for wide spread changes to handle the new instructions that were present on the 7094.[13]

Support for the 1301s produced the biggest changes to the system. These changes had some good aspects and at least one unavoidable one. The good ones included moving from a tape resident system to one that resided on disk and, for users, sequential and random access to temporary files on disk. The unavoidable was that the changes required more space than was available in the 8K system area. This forced us to restructure that area and to use a storage overlay technique. This was palatable only because the system was now disk-resident.

Up to this point, system maintenance was done by the system itself, using a modified version of the system tape editor UA-STE. As an expedient, maintenance was done by switching to the IBSYS Editor. This was accomplished by incorporating a slightly modified IBSYS as a special purpose subsystem under BESYS.

The user was provided with I/O facilities at the FORTRAN and FAP levels to randomly access records within a file. Also, all the I/O facilities formerly used to access tapes were now made capable of sequentially accessing files on disk. This was a prelude to accessing user files stored on disk that would eventually be added to the system.

Facilities were introduced to make timer-cuts and floating-point traps more flexible. Exits could be set up to user code when these events occurred. For timer-cuts, if a timer exit was requested, a 10-second time extension would be granted and a one shot exit invoked. The exit procedure could wrap up processing and then go to ENDJOB. If for some reason the extension also expired, the job would simply be cut. For floating-point traps, any user-provided exit would simply override normal system handling. In addition, the user could request dumps conditioned on floating-point traps.

BESYS-5 was installed in May 1963. After the normal shakedown period, development work resumed aimed at fulfilling disk support as well as providing additional facilities of an innovative nature. However, as a result of the trauma of running out of space in the system area of memory and resorting to the overlay scheme, the development effort split in two. One effort, unofficially dubbed BESYS-6, since it was going to be the "next" system, set out to replace the fixed overlay scheme with a dynamic relocatable approach. The other effort, eventually called BESYS-7, continued the BESYS-5 approach concentrating on completing the promised disk support.

BESYS-6

The intent of the so-called BESYS-6 system was to organize the 8K system area of memory so that it consisted of a static nucleus of minimal size and a dynamic area into which relocatable portions of the system would be dynamically loaded as dictated by a job's processing. There was never any doubt that this was an ambitious approach that had a certain aesthetic appeal. That the associated effort would ever produce a production quality system, however, was in doubt.

Several practical factors were working against this approach. Foremost was performance. BESYS-5 had set a de facto performance level that would be difficult to beat. The overlay scheme was reasonably thought out and was designed to be optimal with respect to the way jobs normally used the system. The dynamic approach inherently had a higher overhead having to deal with module relocation and core fragmentation.

There would be no pending hardware upgrades to boost performance that could be traded for the increased overhead, and there was no hardware assist that could be used to reduce the overhead. The handwriting was on the wall for the 709X machines; attention had already shifted to the next generation of machines.

Finally, the promised upgrades for the disk support were ready and in demand. They would be offered in an edition of BESYS called BESYS-7.

BESYS-7

BESYS-7 was put into service in May, 1964. It delivered the promised additional disk storage support, support for private libraries, a user facility for input source switching, and some rudimentary terminal support.

User Disk Storage

When BESYS-5 was introduced, disk storage space was divided into three categories: permanent, semi-permanent, and temporary. The permanent space was used for system residence, and the temporary space was available for use by users for scratch files. With BESYS-7, provision was made to allocate semi-permanent space to users. Space from the semi-permanent category, called a key area, was allocated to a user on application to the computer center, and a name, called a key, was assigned to access it.

A facility called REVISE enabled users to manage their key areas and to dynamically create, name, and allocate the space to files. Such files could be opened using their key and file names and accessed using any system I/O routine.

As rudimentary as this file system was, it proved to be immensely popular. And surprisingly for a random access facility, it was primarily used to store many modest-sized sequential files. Development groups that used specialized sets of tools found the file system particularly effective because it permitted the tools to be centralized, made readily available to their users, and easy to maintain.

Source Switching and Private Libraries

Two system facilities added to the effectiveness of the disk file system. The first was a standard source switching capability, and the second was the ability to maintain and use private libraries of relocatable subroutines. Source switching could be performed at the control card or program level. It enabled users to redirect the system's standard input to any file. Almost overnight, drawers of job decks quietly slipped into the file system and suddenly cards took a step towards obsolescence. Private libraries that could be searched by the system's loader abetted this shift and made the life of developers of specialized tools even easier.

Experimental Terminals

In 1964, an experimental PB-250-based intelligent graphics terminal, developed by E. N. Pinson, and several typewriter-like terminals with local buffer storage were linked to the 709X systems and made to "interact" with the system at the job level. Although they showed that users could interact directly with the system, the job processing nature of the system precluded the type of interaction taken for granted today.

The End of The Line

BESYS-7 was the last of the systems we produced for the 709X machines. With the next generation hardware waiting in the wings, the pace of system development dwindled. Attention turned to the development of Multics and developments in the TSS/360 and OS/360 world. Optimistically, it was believed that one or the other of these systems would assume the work load being handled by BESYS.

By 1967, this optimism faded as it became clear that such a step would not be cost-effective. The investment in existing software was too great, and the conversion costs were too high for a flash cut to work. Instead a proposal by the author to emulate BESYS-7 on the System/360 as a means of smoothing the transition to the next generation machines and spreading out the conversion costs, gained favor.[14]

BE90 EMULATOR FOR BESYS-7

In 1967, the proposal to emulate BESYS-7 turned to action. A team led by the author and including F. T. Grampp and eventually G. J. Hansen was formed to act on it.

IBM had already produced an emulator for the 709X machines, called EM90. However, it did not support disk storage devices and, therefore, was not suitable for BESYS-7 emulation. Our approach was to develop an emulator, called BE90, by combining EM90's CPU emulation modules with new routines designed to meet BESYS-7's I/O requirements. In addition, direct interfaces to BESYS-7 and ASP were developed so that a mixture of BESYS and OS jobs could be processed within the same OS/360-ASP environment.

BESYS-7 itself was streamlined slightly, dropping most of the code supporting the Tape Control System in favor of set

up information supplied by ASP. All other BESYS functions were supported and no user job needed modification.

BE90 was put into service in March, 1968, and within a month the plug was pulled on two 7094s. The last of the 709X machines would be retired from Bell Labs in March, 1969, and BESYS would continue to be used under emulation until February, 1971.

THE LAST CURTAIN CALL

One of the last programs that BESYS would ever run involved a humanitarian effort to help a boy being treated for Hodgkin's disease. K. Knowlton, a Bell Labs pioneer in computer graphics, had helped develop a technique for generating movies visualizing the treatment of deep body cancer by radiation. Since the computer programs for that purpose only ran under BESYS at the time, the system was fired up one more time under BE90 to process the patient's data. The results produced would determine the proper radiation doses and what vital organs and other parts of the body should get the radiation.[15]

One couldn't have asked for a more fitting end to BESYS's long and distinguished record of service.

CONCLUDING REMARKS

BESYS contributions are hard to quantify. Certainly it helped thousands of scientists and engineers gain more insight into their work as well as provide them the means to obtain the results they required. The author is pleased to have been a principal contributor to the development of BESYS-3 through BESYS-7.

Unlike IBM's IBSYS and OS, it didn't attempt to be all things to all people. Instead, it took a series of machines that had potential but were complex and difficult to use and provided a system that transformed them into efficient and effective tools.

Some like to say that BESYS influenced UNIX; but in practice, no more than L1 and L2 influenced BESYS. For their time, they are all good systems that marched off at right angles to one another sharing only the common bond and spirit of the people that work in the environment established by Bell Labs.

REFERENCES

1. *Encyclopedia of Computer Science and Technology,* Vol. 3, New York: Marcel Dekker, p. 210.
2. Holbrook, Bernard D. and W. Stanley Brown. "A History of Computing Research at Bell Laboratories (1937–1975)." *Computing Science Technical Report No. 99,* Bell Laboratories, 1982, p. 15.
3. Wolontis, V. M. *Bell Laboratories Record,* 52 (1974) 1, p. 18.
4. Wolontis, V. M. "A Complete Floating-Decimal Interpretive System for the IBM 650 Magnetic Drum Calculator." *IBM Technical Newsletter,* No. 11, March 1956.
5. Hamming, R. W. and R. A. Weiss, "General Purpose System." unpublished internal memorandum, Bell Laboratories, September 14, 1956.
6. McLaughlin, Richard A. "The IBM 704: 36-Bit FLoating-Point Money Maker." *Datamation,* 21 (1975) 8, p. 45.
7. Mealy, G. H. "704 Input-Output System: BE SYS 1." unpublished internal memorandum, Bell Laboratories, October 16, 1957.
8. Mealy, G. H. "704 Input-Output System—BE SYS 1 and 2." unpublished internal memorandum, Bell Laboratories, February 13, 1958.
9. Hansen, G. J., W. L. Mammel, and G. H. Mealy. "704 Input-Output and Monitor system—BE SYS 2." unpublished internal memorandum, Bell Laboratories, May 22, 1959.
10. Holbrook, B. D. and W. S. Brown. "Bell Laboratories and the Computer from the Late 30's to the Middle 60's." unpublished internal memorandum, Bell Laboratories, June 25, 1975, p. 54.
11. Eastwood, D. E. and M. D. McIlroy. "Macro Compiler Modification of SAP." unpublished internal memorandum, Bell Laboratories, September, 1959.
12. Drummond, R. E. and G. J. Hansen. "BE-SYS-4 Release Description." unpublished internal information bulletin, Bell Laboratories, February, 1962.
13. Cutler, M. R. "General Description of BESYS5." unpublished internal memorandum, Bell Laboratories, February 18, 1964.
14. Drummond, R. E. "Emulation of BESYS-7 on The SYSTEM/360, Model 65." unpublished internal memorandum, Bell Laboratories, April 3, 1967.
15. "Movies Help Radiation Treatment." *Bell Labs News,* May 21, 1971.