The ADEPT-50 time-sharing system

by R. R. LINDE and C. WEISSMAN

System Development Corporation
Santa Monica, California

and

C. E. FOX

King Resources Company
Los Angeles, California

INTRODUCTION

In the past decade, many computer systems intended for operational use by large military and governmental organizations have been "custom made" to meet the needs of the particular operational situation for which they were intended. In recent years, however, there has been a growing realization that this design approach is not the best method for long term system development. Rather, the development of general purpose systems has been promoted that provide a broad, general base on which to configure new systems. The concepts of time-sharing and general-purpose data management have been under development for several years, particularly in university or research settings.[1,2,3] These methods of computer usage have been tested, evaluated, and refined to the point where today they are ready to be exploited by a broad user community.

Work on the Advanced Development Prototype (ADP) contract was begun in January 1967 for the purpose of demonstrating—in an operational environment—the potential of automatic information-handling made possible by recent advances in computer technology, particularly advances in time-sharing executives and general-purpose data management techniques. The result of this work is a large-scale, multi-purpose system known as ADEPT, which operates on IBM system 360 computers.*

The entire ADEPT system is now being used at four field installations in the Washington, D. C. area, as well as at SDC in Santa Monica. The system was installed at the National Military Command System Support Center in May 1968, at the Air Force Command Post in August 1968, and at two other government agencies in January 1969. These four field sites collectively run ADEPT from 80 to 100 hours per week, providing a total of some 2000 terminal hours of time-sharing service monthly to their users.

The ADEPT system consists of three major components: a time-sharing executive; a data management system adapted from SDC's Time-Shared Data Management System (TDMS) described by Bleier,[4] and a programmer's package. This paper deals exclusively with the ADEPT Time-Sharing Executive, and particularly with the more novel aspects of its architecture and construction. Before examining these aspects it will be instructive if we review the basic design and hardware configuration of the system.

A general purpose operating system

The ADEPT executive is a general-purpose time-

* Development of ADEPT was supported in part by the Advanced Research Projects Agency of the Department of Defense.

sharing system. The system operates on a 360 Model 50 with approximately 260,000 bytes of core memory, 4 million bytes of drum memory, and over 250 million bytes of disc memory, shown graphically in Figure 1 and schematically in the appendix. With this machine configuration, ADEPT is designed to provide responsive on-line interactive service, as well as background service to approximately 10 concurrent user jobs. It handles a wide variety of different, independent application programs, and supports the use of large random-access data files. The design—basically a swapping system—provides for flexibility and expansion of system functions, and growth to more powerful models in the 360 family.

ADEPT functions both as a batch processor (whereby jobs are accumulated and fed to the CPU for operation one by one) and as an interactive, on-line system (in which the user controls his job directly in real time simply by typing console requests).

Viewed as a batch system, ADEPT allows jobs to be submitted to console operators or submitted from consoles via remote batch commands (remote job entry). In either case, jobs are "stacked" for execution by ADEPT in a first-in/first-out order. The stack is serviced by ADEPT as a background task, subject to the priorities of the installation and the demands of "foreground" interactive users. Viewed as an interactive system, ADEPT allows the user to work with a typewriter, allowing computer-user dialog in real time. Via ADEPT console commands, the user identifies himself, his programs, and his data files, and selectively controls the sequence and extent of operation of his job in an ad lib manner. A prime advantage of the interactive use of ADEPT is that the system provides an extendable library of service programs that permit the user to edit data files, compile or assemble programs, debug and eliminate program errors, and generally manage large data bases in a responsive on-line manner.

System architecture

The architecture of the ADEPT executive is that of the "kernel and the shell". The "kernel," referred to as the Basic Executive (BASEX), handles the major problems of allocating and scheduling hardware resources. It is small enough to be permanently resident in low core memory, permitting rapid response to urgent tasks, e.g., interrupt control, memory allocation, and input/output traffic. The "shell," referred to as the Extended Executive (EXEX), provides the interface between the user's application program and the "kernel". It contains those non-urgent, large-

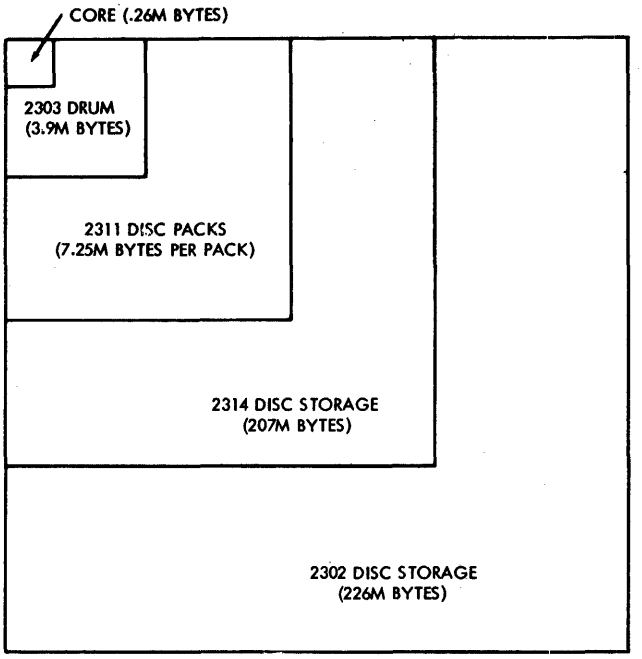

Figure 1—Relative capacity of various ADEPT direct-access storage media available in less than 0.2 seconds. The initial system that operates at SDC utilizes core, 2303 drum, 2311 and 2314 disc packs, and 2302 disc storage. The NMCSSC system utilizes 2314 disc storage in lieu of 2311 or 2302 discs. The architecture of the ADEPT executive is such that it permits any combination of the above types of disc storage in varying amounts

task extensions of the basic "kernel" processes that are user-oriented rather than hardware-oriented; they may, therefore, be scheduled and swapped.

The version of the ADEPT time-sharing system thus far developed has multiple levels of control beyond the two-level "kernel-shell" structure—i.e., it can be thought of figuratively as an "onion skin". Figure 2 shows these relationships graphically.

Beyond EXEX, "object systems" may exist as subsystems of ADEPT (developed by the user community without modification to EXEX or BASEX), thus further distributing and controlling the system resources for the object programs that form still another level of the system. The design ideas embodied in ADEPT parallel those of Dijkstra,[5] Corbato,[6] and Lampson,[7] but differ in techniques of implementation.

The ADEPT Basic Executive operates in the lower quarter of memory, thereby providing three quarters of memory for user programs. With the current H core configuration, ADEPT preempts the first 65,000 bytes of core memory, the bulk of which is dedicated to BASEX; EXEX must then operate in user memory

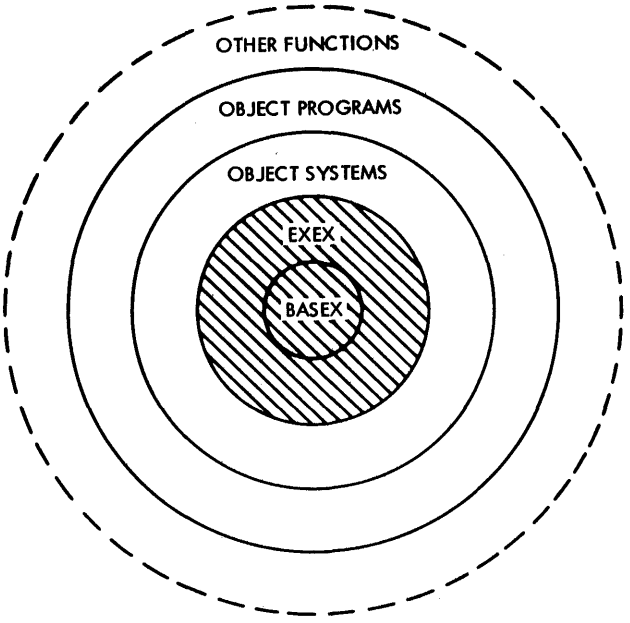

Figure 2—Multiple levels of control in ADEPT

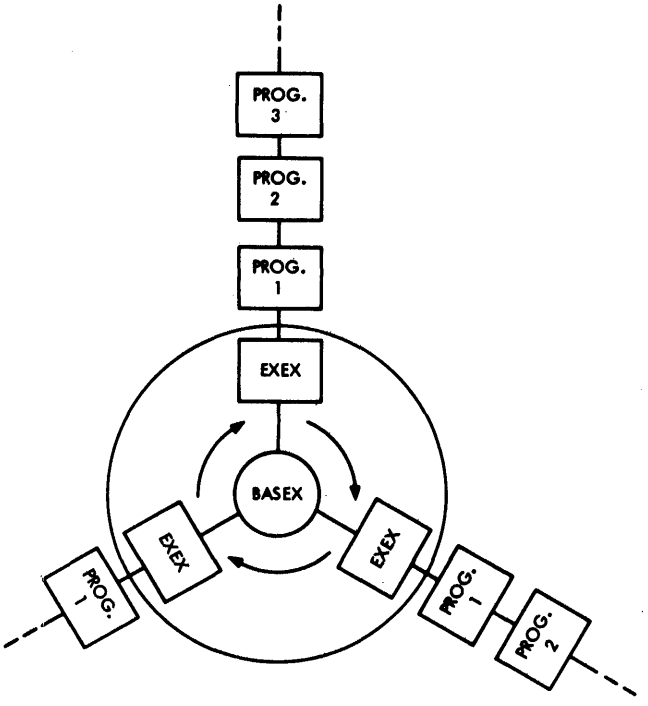

Figure 3—Simple commutation of users programs. This figure illustrates the relationship between user's programs' EXEX and BASEX. Each spoke represents a user's job, with his EXEX providing the interface between BASEX and the hardware resources. The maximum number of interactive job the IBM 360/50H configuration is ten.

in a fashion similar to user programs. ADEPT is designed to operate itself and user programs as a collection of 4096-byte pages. BASEX is identified as certain pages that are fixed in main storage and that cannot be overlayed or swapped. EXEX and other programs are identified as sets of pages that move dynamically between main storage and swap storage (i.e., drum). It is necessary to maintain considerably more descriptive information about these swappable programs than about BASEX. This descriptive information is carried in a set of system tables that, at any point in time, describe the current state of the system and each program.

ADEPT views the user as a job consisting of some number of programs (up to four for the 360/50H configuration) that were loaded at the user's request. These programs may be independent of one another or, with proper design, different segments of a larger task. Implicitly, EXEX is considered to be one of these programs. To simplify system scheduling, communication, and control, only one program in the user's set may be active (eligible to run) at a time. When ADEPT scheduling determines that a job may be serviced, the current job in core is saved on swap storage, and the active program of the next job is brought into core from swap storage and executed for a maximum period of time, called a quantum. The process then repeats for other jobs. Figures 3 and 4 schematically depict these relationships.

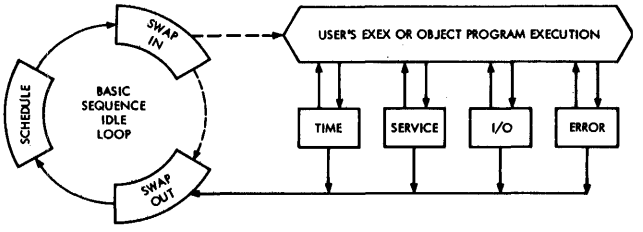

Figure 4—ADEPT's basic sequence of operation. This figure shows the basic operating system cycle: idle loop is interrupted by an external interrupt (an activity request); a program is scheduled, swapped into core from the drum, and executed escape from the execution phase occurs when quantum termination condition (e.g., time expiration, service or I/O call, error condition) is met; the program is then swapped out and control is returned to the idle loop (if no other programs are eligible to be scheduled).

Basic executive (BASEX)

Table I lists the BASEX components and their general functions as of the eighth and latest executive release. These basic system components form an integrated, non-reentrant, non-relocatable, perma-

nently-resident, core memory package 16 pages long (each page is 4096 bytes). They are invoked by hardware interrupts in response to service requests by users of terminals and their programs. Note the division of input/output control into cataloged (SPAM and IOS), terminal (TWRI), and drum (BXEC) activities to permit local optimization for improved system performance.

TABLE I—Basic executive components

Component	*Function*
ALLOC	Drum and core memory allocation.
BXBUG	Debugger for executive programs.
BXEC	Basic sequence and swap control.
BXECSVC	SVC handlers for WAIT, TIME, DEVICE, STOP AND DISMISS calls.
EXEX	Linkage routines for EXEX (BASEX/EXEX interfaces); also services commands DIALOFF, DIALON.
INTRUP	First-level interrupt control.
IOS	Channel-program level input/output supervisory control.
RECORD	Records SVC, interrupt activity in BASEX.
SKED	Scheduler.
SPAM	Input/output access methods to cataloged storage.
TWRI	Terminal input/output control.
System Tables	Resident system data areas for communication table (COMTAB), logged-in user's table (JOB), loaded programs table (PQU), drum and core status tables (DSTAT, CSTAT), and a variety of other tables.

Extended executive (EXEX)

Unlike the tight, closed package of integrated BASEX components, EXEX is a loose, open-ended collection of semiautonomous programs. Table II lists this collection of programs. EXEX is treated by BASEX as a user program, with certain privileges, and each user is given his own "copy" of the EXEX. It is transparent to the user that EXEX is reentrant

TABLE II—Extended executive components

Component	*Function*
AUDIT	Maintains a real-time recording of all security transactions as an accountability log.
BMON	Batch monitor for control of background job execution.
CAT	Cataloger for file storage access control; also services FORGET command.
DTD	Transfers recording information from drum to disc.
DBUG	Debugger for non-executive (user) programs.
LOGIN	User authentication and job creation.
SERVIS	Library of service commands that are reentrant, interruptible and scheduled: APPEND, CHANGE, CREATE, CYLS, DELETE, DRIVES, INIT, LISTF, LISTU, LOAD, LOADD, LOAD and GO, OVERLAY, REPLACE, RESTORE, RESTORED, SAVE, SEARCH, VARYOFF, VARYON.
RUN	Remote batch job submission control servicing commands RUN and CANCEL.
XXTOO	Library of small, fast, executive service commands: CPU, BGO, BQUIT, BSTOP, DIAL, DRUMS, GO, LOGOUT, QUIT, RESTART, SKED, SKEDOFF, STATUS, STOP, TIME, USERS.
SYSDEF	Defines input/output hardware configuration at time of system start up.
SYSLOG	Defines authorized user/terminal security profiles at time of system start up.
TEST	Initializes system tables at time of system start up.
SYSDATA	Non-resident, shared, system data table for dial messages and other common data, e.g., lists of all logged-in users; other non-resident, job-specific tables also exist, e.g., job environment page, push-down list data page.

and is being shared with other users, except for its data space. Each job has its own "machine state" tables saved in its unique set of environment pages. This structure permits flexible modification and orderly system expansion in a modular fashion. EXEX is always scheduled in the same way as other user programs.

Though EXEX components are, in large part, non-self-modifying reentrant routines and thus, could at small cost, be relocatable; neither user programs nor EXEX components are relocated between swaps. The lack of any mapping hardware on the IBM 360/50 and the design goal and knowledge that most user programs would be of maximum size made unnecessary a software provision to relocate programs dynamically. User programs may be relocated once at load time, however.

Communication and control techniques used in ADEPT

Communication is the generic term used to cover those services that permit two (or more) programs to intercommunicate, be they system program, user program, or both. From this communication vantage point we shall examine the connective mechanism used between the Basic and Extended Executives; the techniques that allow components within the EXEX to make use of one another; and the system design that permits an object program to control its own behavior as well as to communicate with the system and with other object programs.

The ADEPT job or process

Before we discuss the system mechanics, let us examine how the system treats each user logically. A user in the system is assigned a job number. Each job in the system may be viewed as a separate *process*, and each process is, by definition, independent of all other processes running on the machine. A process— or job— is not a program. It is the logical entity for the execution of a program on the physical processor, and it may contain as many as four separate programs. A program consists of the set of machine instructions swapped into the processor for execution, and the Extended Executive is one of these programs.

The ADEPT executive requires a large number of system tables to permit Basic and Extended Executive communication. Conceptually, the use of descriptive tables defining the condition of a user's process is analogous to the state vector (or state word) discussed by Lampson and Saltzer.[8,9] That is, the collection of information contained by these tables is sufficient to define an inactive user's process state at any given moment. By resetting the central processor from the state vector, a user's job proceeds from an inactive to an active state as if no interruption had occurred. The state vector contains such items as the program counter, the processor's general registers, the core and drum map of all the programs in the job, and the peripheral storage file data. All of the collective data for each program or task in the process are contained in the state vector.

Basic and extended executive communication

Each ADEPT user (i.e., any person who initiates some activity within the system by typing in commands) is given a job number and assigned an entry in the JOB table. The JOB table contains the system's top-level bookkeeping on user activity. It contains the user's identification, his location, his security clearance, and a pointer to his program queue. Each user is assigned one entry, or JOB, in the table. Associated with each JOB are the one or more programs that the user is running.

Top-level bookkeeping on programs is contained in the Program Queue (PQU) table. Each PQU entry contains a program identification and some (but not all) information that describes that program in terms of its space requirements, its current activity, its scheduling conditions, and its relationship to other programs in the PQU that belong to the same JOB. The detailed descriptive information and the status of each JOB and its programs are carried in the swappable environment space.

The environment pages (there can be as many as four) comprise a number of separate tables that contain such information as the contents of the general registers, the swap storage page numbers where the balance of the program resides, the program map, and lists of all active data files. A single environment page (or pages) is shared by all programs that belong to the same JOB (user). The system design allows for environment page overflow at which time additional pages are assigned dynamically. The environment pages, PQU table, JOB table, and data pages comprise the state vector of the user's job.

To permit storage of "global" system variables, and to allow system components to reference system data that may be periodically relocated, there exists a system communication table, which resides in low core so that it can be referenced without loading a base register.

The IBM 360 supervisor call (SVC) is used exclu-

sively by EXEX components and object programs to request BASEX services. Though additional overhead is incurred in the handling of the attendant interrupt, the centralization of context switching provided is of considerable value in system design, fabrication, and checkout.

Extended executive communication

An EXEX may make use of another EXEX function by use of the SVC call mechanism. To support the recursive EXEX, an additional SVC processing routine is required to manage the different recursive contexts. This routine, called the SVC Dispatcher, processes calls from user and EXEX functions alike, manages a swappable data page, and switches to an interface linkage routine. The data page contains a system communication stack that consists of a program's general registers and the Program Status Word at the time of the SVC. This technique is analogous to the push-down logic of recursive procedure calls found in ALGOL or LISP language systems. The stack provides a convenient means of passing parameters between routines in the EXEX. Since each job has its own unique data page and environment page, EXEX is both recursive and reentrant.

The environment status table (ESTAT) contains the swap and core location for each component in the EXEX and for each program in the job. It resides in the job environment page. When an EXEX service is requested, only that particular EXEX program is brought in from swap storage, rather than the full service library. The interface linkage routine provides this management function; it lies as a link between the SVC Dispatcher and the particular EXEX function. The interface routine picks up necessary work pages for the EXEX component involved and branches to that component after it is brought into core. The interface routine maintains a separate push-down stack of return addresses providing the means for the EXEX component to properly exit and return control to its interface routine and then to the system.

The EXEX component called may make additional EXEX SVC calls before exiting. To provide correct work page allocation during recursive calls, the interface routine also saves the work page core and drum page addresses in the push-down stack. Upon completion of a call, the EXEX component returns to its interface routine; the interface routine releases all allocated work pages to the system and branches to a common unwind procedure.

The unwind procedure, like the SVC Dispatcher, is simply a switching mechanism. It determines, via the stack, whether to return to a still higher level EXEX function, or to turn the EXEX off and exit to the Basic Sequence. This recursive/reentrant control is the most complex portion of ADEPT and is the "glue" that binds BASEX and EXEX together. Figure 5 illustrates the recursive process.

Object program communication

One of the more stringent services required of an operating system is the rapid interchange of large quantities of data between object programs. The interchange of even simple arrays, matrices, and tables via stack parameters or a common file suffers from the inadequacy of limited capacity or extensive I/O time. Many operating systems ignore this requirement, thereby restricting the general-purpose applications. Yet there are solutions to this problem, and one successful technique employed in the ADEPT system is that of "shared memory". Shared memory is achieved by using the basic mechanism for managing reentrancy, namely the program environment page map. Through the ADEPT SHARE Page call, an object program can request that designated pages of another program

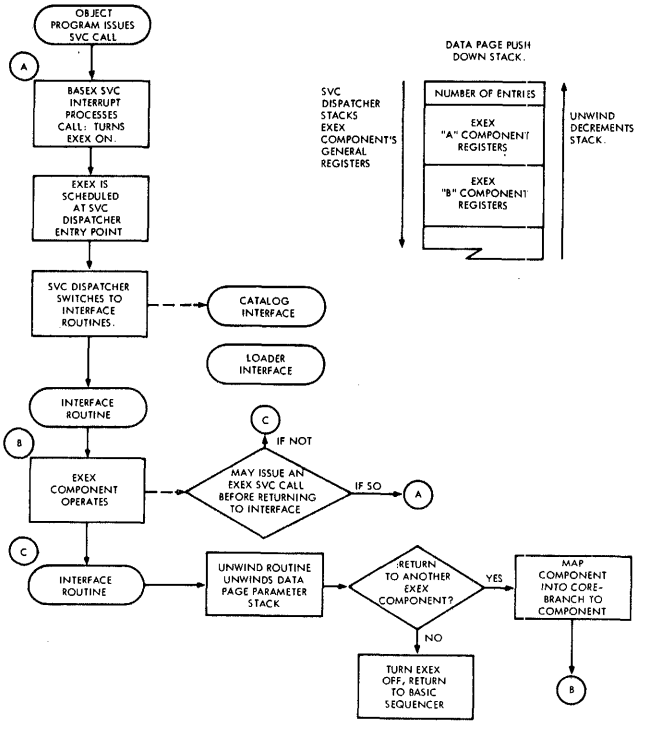

Figure 5—Block diagram of EXEX behavior and control

in the job be added to its map. If core page numbers are passed as parameters in various service calls, whole pages of data may be passed between programs. EXEX and many object programs operating under this system use this method for inter-program communication.

ADEPT operating on the IBM 360/50H restricts its user programs to 46 active core pages. However, by utilizing the GETPAGE call, an object program may acquire up to 128 drum pages and may subsequently activate and deactivate various page sets by utilizing another service call, ACTDEACT (activate/ deactivate). This scheme permits bulk data from disc storage to be placed on drum and operated upon at "swap" speeds. Thus skilled system users can achieve efficient use of time and memory by managing their own "paging". We consider this the best alternative considering the questionable state of other, automatic paging algorithms.[10,11,12,13] Most EXEX components use these calls for just such purposes. For example, the interface routines mentioned above use activate calls to "turn on" called components of the EXEX.

The Allocator component of ADEPT manages the page map for each program. This software map reflects the correspondence between drum and core pages, established initially by the SERVIS (service) component at load time. The Allocator's function is to inventory available core and drum pages by maintaining two resident system tables: one for core, the other for drum. Whenever drum pages are released or obtained, the Allocator updates the page map in the job's environment page. The Allocator processes the SHARE (page), GETPAGE, FREEPAGE, and ACTDEACT calls from EXEX and object programs. SERVIS allows a program at run time to add data pages or to overlay program segments from disc or tape. In so doing, SERVIS makes use of the various Allocator calls.

Simulating console commands

An important attribute of ADEPT time-sharing is that nearly all the functions and services that can be initiated at the user's console can also be called forth within a user's program. A program designer can, for example, build a system of programs, which can operate in batch mode under the control of a program by issuing internal commands in much the same manner as the user sitting at the console. With this approach, the ADEPT batch monitor controls background tasks by simulating user terminal requests. Batch requests can be enqueued by users from any console and then processed in turn by this supervisor function.

Armed interrupts and rescue function

The basic design of ADEPT conveniently provides for processing object program "armed" interrupt calls. This means that an object program is able to conditionally start (wakeup) and stop (sleep) the execution of its own programs, and others as well. The conditions for employing wakeup calls include too much elapsed time, or the occurrence of unpredictable but anticipated events, e g., errors and other program calls. In "arming" these "software-interrupt" conditions by object program calls, the program entry point(s) for the various conditions are specified. When such conditions occur, the operating system transfers to the specified entry point and gives the appropriate condition code. (Note that if we take this call one step further, and permit one object program to arm the software and hardware interrupts of another object program, we have the basic control mechanism necessary to permit the operation of "object systems. necessary to permit the operation of "object systems," i.e., subexecutives—another level in the "onion skin" of ADEPT control.)

User programs interface with the ADEPT system primarily via the supervisor call (SVC) instruction; a secondary interface is provided via the program check interrupt that protects the program and system after various error conditions. The executive design allows user programs to trap all such interfaces with the system via its rescue arming mechanism. This means that one program can trap and get first-level control of all occurrences of SVC's and program checks within a single job. This mechanism also means, then, that the responsibility and meaning for these interfaces can be redefined at the user program level.

As of this writing, this mechanism is being employed to construct object systems for an improved batch monitor, an interface for the proposed ARPA Network,[14] and to experiment with automatic translators for compatibility with other operating systems. Other uses include improvements in program recovery in a variety of user tools, e.g., compiler diagnostics.

Resource allocation, access, and management

ADEPT system design, of course, includes a complete set of resource controls that monitor secondary storage devices.

The cataloger

The Cataloger, an EXEX component, is functionally analogous to the core/drum Allocator, but is used for devices accessible by user programs. It maintains an inventory of all assignable storage devices, assigns unused storage on the devices, maintains descriptions of the files placed on these devices, controls access to these files, and—upon authorized request—deletes any file. Specifically, the Cataloger:

- Assigns storage on 2302, 2311 and 2314 discs.
- Assigns tape drives.
- Locates an inventoried file by its name and certain qualifiers that uniquely identify the file.
- Issues tape or disc pack mounting instructions to the operator when necessary.
- Verifies the mounting of labeled volumes.
- Passes descriptive information to the user program opening a file.
- Allows the user of a file to request more storage for the file.
- Denies unauthorized users access to files.
- Returns assigned storage to available storage whenever a file is deleted.
- Maintains a table of contents on each disc volume.

As the largest single component of the ADEPT Eexcutive (65,000 bytes), the Cataloger was written in a new, experimental programming language called MOL-360 (Machine-Oriented Language for the 360).[15] It is a "higher-level machine language" developed under an ARPA-sponsored SDC research project on metacompilers. It resolved the dilemma involving our desire for higher-level source language and our need to achieve flexibility with machine code. The Cataloger design and checkout, enhanced by the use of MOL-360, showed simultaneously the validity of MOL compilers for difficult machine-dependent programming.

The SPAM component

SPAM is a BASEX component that permits symbolic, user-oriented I/O. It can be viewed as a special-purpose compiler that compiles symbolic user program I/O calls into 360 channel programs, and delivers them to the Input/Output Supervisor (IOS) for execution via the EXCP (execute channel program) call. The results of EXCP for the call are "interpreted" by SPAM and returned to the user program as status information. As such, SPAM represents a more symbolic I/O capability than the EXCP level. It provides a relatively simple method for executing the operations of reading, writing, altering, searching for, and positioning records within ADEPT cataloged and controlled disc-based and tape-based file structures.

Resource management

As of this writing, the computer operator has a set of commands at his disposal that allow him to control the system resources. Various privileged on-line commands enable him to monitor the terminal activities of system users and to control assignment and availability of storage devices. However, there is an increasing need for a "manager" to be given more latitude in dynamically controlling the system resources and observing the status of system users, particularly because ADEPT was designed to handle sensitive information in classified government and military facilities. To meet these objectives, a design effort is under way that gives the computer operator system-manager status, with the ability to observe and control the actions of system users. The result will be a program that encompasses some of the management techniques reported by Linde and Chaney[16] tailored to present needs.

Swapping and scheduling user programs

Most of the programs that run under ADEPT occupy all of the core memory that is not used by the resident Basic Executive (46 pages on the 360/50H). If the set of needed pages could be reduced considerable reduction in swap overhead could be expected. One way to achieve this is to mark for swap-out only those pages that were changed during program execution. The hardware needed to automatically mark changed pages is unavailable for the 360/50; however, through use of the store-protect feature on the Model 50, ADEPT software can simulate the effect and produce noteworthy savings in swap time.

Page marking

Whenever a user program is swapped into core, its pages are set in a read-only condition. As the program executes, it periodically attempts to store data (write) in its write-protected pages. The resulting interrupt is fielded by the system. After satisfying itself that the store is legal for the program, the executive marks the target page as "written," turns off write-protect

for that page, and resumes the program's execution. The situation repeats for each additional page written. At the completion of the program's time slice, the swapper has a map of all the program pages that were changed (implied in the storage keys with no write protection). Only the changed pages are swapped out of core. Measurement of this scheme shows that about 20 percent of the pages are changed; hence, for every five pages swapped in, only one need be swapped out, for a total swap of six pages, rather than the full swap of ten pages (five in, five out). The scheme makes the drum appear to be 40 percent faster.

The use of the storage protection keys is based on the functional status of each page rather than on some user identity. User programs always run with a program status word key of one, and the bits in the storage key associated with the programs start out at zero. After a page has been initially changed, its key is set to one also. The other bits in the key are used to indicate: first, a page is transient, not yet completely moved to or from swap storage; second, a page is unavailable, i.e., it belongs to someone else; third, a page is locked and cannot be swapped or changed; and finally, a page is fetch-protected because it may contain sensitive information.

Scheduling algorithm

The scheduling algorithm provides for three levels of scheduling. Jobs that are in a "terminal I/O complete" state get first preference in the schedule. Jobs in the second level, or background queue, are run if there are no level-one jobs to run. A job is placed in level two when the two-second quantum clock alarm terminates its operation two consecutive times. Compute and I/O-bound programs are treated alike. A level-two job—when allowed to run—is given quantum interval equal to the basic quantum time multiplied by the scheduling level (i.e., 2 sec $\times$ 2 = 4 sec). However, a level-two background job may be preempted after two seconds for terminal I/O. Any operation a level-two job makes that terminates its quantum prematurely will return the job to a level-one status. The batch monitor job is run when the first two queues are empty. User programs may be written to overlap execution and I/O activity. Our choice of scheduling parameters for quantum size, and number of service levels was selected empirically and as a result of prior experience.[17]

A command SKED, which is limited to the operator's terminal, has the effect of forcing top priority for a job (the job stays at level one all the time). Only one job may run in this privileged scheduling state at a time.

Pervasive security controls

Integrated throughout the ADEPT executive are software controls for safeguarding security-sensitive information. The conceptual framework is based upon four "security objects": user, terminal, file, and job. Each of these security objects is formally identified in the system and is also described by a security profile triplet: Authority (e.g., TOP SECRET, SECRET), Need-to-Know Franchise, and Special Category (e.g., EYES ONLY, CRYPTO). At system initialization time, user and terminal security profiles are established by security officers via the system component SYSLOG. SYSLOG also permits the association of up to 64 passwords with each user. At LOGIN time, a user identifies himself by his unique name, up to 12 characters, and enters his private password to authenticate his identity. The LOGIN component of ADEPT validates the user and dynamically derives the security profile for the user's job as a complex function of the user and terminal security profiles. The job security profile is used subsequently as a set of "keys," used when access is made to ADEPT files. The file security profile is the "lock" and is under control of the file subsystem.

File access Need-to-Know is permitted for Private, Semi-Private, and Public use. With the CREATE command, a list of authorized users and the extent of their access authorization (i.e., read-only, write-only, read and write) can be established easily for Semi-Private files. Newly created files are automatically classified with the job's "high water mark" security triplet—a cumulative security profile history of the security of files referenced by the job. Through judicious use of the CHANGE command, these properties may be altered by the owner of the file.

Security controls are also involved in the control of classified memory residue. Software and hardware memory protection is extensively used. Software memory protection is achieved by interpretive, legality checking of memory bounds for I/O buffer transfers, legality checking of device addresses for unauthorized hardware access, and checks of other user program attempts to seduce the operating system into violating security controls.

The hardware protection keys are used to fetch-protect all address space outside the user program and data area. Also, newly allocated space to user programs is zeroed out to avoid classified memory residue.

for that page, and resumes the program's execution. The situation repeats for each additional page written. At the completion of the program's time slice, the swapper has a map of all the program pages that were changed (implied in the storage keys with no write protection). Only the changed pages are swapped out of core. Measurement of this scheme shows that about 20 percent of the pages are changed; hence, for every five pages swapped in, only one need be swapped out, for a total swap of six pages, rather than the full swap of ten pages (five in, five out). The scheme makes the drum appear to be 40 percent faster.

The use of the storage protection keys is based on the functional status of each page rather than on some user identity. User programs always run with a program status word key of one, and the bits in the storage key associated with the programs start out at zero. After a page has been initially changed, its key is set to one also. The other bits in the key are used to indicate: first, a page is transient, not yet completely moved to or from swap storage; second, a page is unavailable, i.e., it belongs to someone else; third, a page is locked and cannot be swapped or changed; and finally, a page is fetch-protected because it may contain sensitive information.

Scheduling algorithm

The scheduling algorithm provides for three levels of scheduling. Jobs that are in a "terminal I/0 complete" state get first preference in the schedule. Jobs in the second level, or background queue, are run if there are no level-one jobs to run. A job is placed in level two when the two-second quantum clock alarm terminates its operation two consecutive times. Compute and I/O-bound programs are treated alike. A level-two job—when allowed to run—is given quantum interval equal to the basic quantum time multiplied by the scheduling level (i.e., 2 sec $\times$ 2 = 4 sec). However, a level-two background job may be preempted after two seconds for terminal I/O. Any operation a level-two job makes that terminates its quantum prematurely will return the job to a level-one status. The batch monitor job is run when the first two queues are empty. User programs may be written to overlap execution and I/O activity. Our choice of scheduling parameters for quantum size, and number of service levels was selected empirically and as a result of prior experience.[17]

A command SKED, which is limited to the operator's terminal, has the effect of forcing top priority for a job (the job stays at level one all the time). Only one job may run in this privileged scheduling state at a time.

Pervasive security controls

Integrated throughout the ADEPT executive are software controls for safeguarding security-sensitive information. The conceptual framework is based upon four "security objects": user, terminal, file, and job. Each of these security objects is formally identified in the system and is also described by a security profile triplet: Authority (e.g., TOP SECRET, SECRET), Need-to-Know Franchise, and Special Category (e.g., EYES ONLY, CRYPTO). At system initialization time, user and terminal security profiles are established by security officers via the system component SYSLOG. SYSLOG also permits the association of up to 64 passwords with each user. At LOGIN time, a user identifies himself by his unique name, up to 12 characters, and enters his private password to authenticate his identity. The LOGIN component of ADEPT validates the user and dynamically derives the security profile for the user's job as a complex function of the user and terminal security profiles. The job security profile is used subsequently as a set of "keys," used when access is made to ADEPT files. The file security profile is the "lock" and is under control of the file subsystem.

File access Need-to-Know is permitted for Private, Semi-Private, and Public use. With the CREATE command, a list of authorized users and the extent of their access authorization (i.e., read-only, write-only, read and write) can be established easily for Semi-Private files. Newly created files are automatically classified with the job's "high water mark" security triplet—a cumulative security profile history of the security of files referenced by the job. Through judicious use of the CHANGE command, these properties may be altered by the owner of the file.

Security controls are also involved in the control of classified memory residue. Software and hardware memory protection is extensively used. Software memory protection is achieved by interpretive, legality checking of memory bounds for I/O buffer transfers, legality checking of device addresses for unauthorized hardware access, and checks of other user program attempts to seduce the operating system into violating security controls.

The hardware protection keys are used to fetch-protect all address space outside the user program and data area. Also, newly allocated space to user programs is zeroed out to avoid classified memory residue.

Typically, the complete system reaches "on the air" status in less than a minute.

System instrumentation

Many of the parameters built into the scheduling and swapping of early ADEPT versions were based upon empirical knowledge. The latest versions of the Basic and Extended Executives include routines to record system performance, reliability, and security locks.

Built into the BASEX is a routine to measure the overall and the detailed system performance.[20] Such factors as the number of users, file usage, hardware and software errors, and page transaction response time are recorded on unused portions of the 2303 drum. These measurements provide a better understanding of the system under a variety of inputs and give the designers insight into how the hardware and software components of the system affect the performance of the human user.

An AUDIT program was made part of the EXEX to record the security interaction of terminals, users, and files. AUDIT records EXEX activity in the areas of LOGIN, LOGOUT, and File Manipulation. This routine strengthens the security safeguards of the executive. Specific items that are recorded involve: type of event, user identification, user account number, job security, device identification, time of event, file identification, file security and event success. In addition, this routine provides accounting information and is used as a means of debugging the security locks of new system releases.

In addition to the BASEX recording function, several object programs have been written that simulate various modes of user activity and provide controlled job distributions. These programs, called "benchmarks," run under controlled conditions and enhance the means of improving system performance and throughput, as described elsewhere by Karush.[21] The programs are designed to gather performance measures on the major routines of the executive and have been of considerable help in system "tuning," because they reflect the effect of coding and design changes to various system routines. The routines in the executive that are of primary concern are the swapper, the scheduer, the terminal read/write package, and the interrupt handling processes. Attempts are being made to design a set of benchmarks that represent a typical job mix. However, we are primarily interested in measuring the performance of our system against various modifications of itself and in measuring its behavior with respect to different job mixes.

SUMMARY

The ADEPT executive is a second-generation, general-purpose, time-sharing system designed for IBM 360 computers. Unlike the monolithic systems of the past,[1,2] it is structured in modular fashion, employing distributed executive design techniques that have permitted evolutionary development. This design has not only produced a flexible executive system but has given the user the same facilities used by the executive for controlling the behavior of his programs. ADEPT's security aspects are unique in the industry, and the testing and fabrication methods employ a number of novel approaches to system checkout that contribute to its operational reliability.

It is important to note that this system deals particularly well with size limitation problems of very large files and very large programs. The provisions made for multiple programs per job, active/inactive page status for programs larger than core size, page sharing between programs, common file access across programs within jobs, and the commitment of considerable space to active file environment tables (up to four pages worth) contribute to this success. Nevertheless, all these capabilities are designed to handle the smaller entities as well. We feel ADEPT-50 is a significant contribution to the technology of general-purpose time-sharing.

ACKNOWLEDGMENTS

We would like to express our appreciation for the dedicated efforts of some very *adept* individuals who participated in the design and building of this time-sharing system. Our thanks go to Mr. Salvador Aranda, Mr. Peter Baker, Mrs. Martha Bleier, Mr. Arnold Karush, Mrs. Patricia Kribs, Mr. Reginald Martin, Mr. Alexander Tschekaloff and all the others who have followed their lead.

REFERENCES

1 P CRISMAN editor
The compatible time-sharing system: A programmer's guide
MIT Press Cambridge Mass 1965

2 J SCHWARTZ et al
A general-purpose time-sharing system
Proc SJCC Vol 25 1964 397-411 Spartan Books Baltimore

3 E W FRANKS
A data management system for time-shared file-processing using a cross-index file and self-defining entries
AFIPS Proc Vol 28 1966 79-86 Also available as SDC document SP-2248 21 April 1966

4 R E BLEIER
Treating hierarchical data structures in the SDC time-shared data management system (TDMS)
Proc 22nd Nat ACM Conf Thompson Book Co 1967 41-49
5 E W DIJKSTRA
The structure of T.H.E. multi-programming system
C A C M Vol 11 No 5 May 1968
6 F J CORBATO V A VYSSOTSKY
Introduction and overview of the multics system
Proc FJCC Nov 30 1965 Las Vegas Nevada
7 B W LAMPSON
Time-sharing system reference manual
Working Doc Univ of Calif Doc No 30.1030
Sept 1965 Dec 1965
8 B W LAMPSON
A scheduling philosophy for multi-processing systems
C A C M Vol 11 No 5 May 1968
9 J H SALTZER
Traffic control in a multiplexed computer system
MAC-TR-30 thesis MIT Press July 1966
10 G H FINE et al
Dynamic program behavior under paging
Proc ACM 1966 223-228 Thompson Book Co Wash D C
11 E G COFFMAN L C VARIAN
Further experimental data on the behavior of programs in a paging environment
C A C M Vol 11 No 7 July 1968 471-474
12 L A BELADY
A study of replacement algorithms for a virtual storage computer
IBM Systems Journal Vol 5 No 2 1966
13 R W O'NEIL
Experience using a time-shared multi-programing system with dynamic address relocation hardware
Proc SJCC 1967 Vol 30 611-627 Thompson Book Co Washington D C
14 L G ROBERTS
Multiple computer networks and intercomputer networks and intercomputer communication
ACM Symposium on Operating System Principles Oct 1-4 1967 Gatlinburg Tenn
15 E BOOK D C SCHORRE S J SHERMAN
Users manual for MOL-360
SCC Doc TM-3086/003/01
16 R R LINDE P E CHANEY
Operational management of time-sharing systems
Proc ACM 1966 149-159
17 P V McISSAC
Job descriptions and scheduling in the SDC Q-32 time-sharing system
SDC Doc TM-2996 June 1966 28
18 C WEISSMAN
Security controls in the ADEPT-50 time-sharing system
AFIPS Proc FJCC Vol 35 1969
19 W A BERNSTEIN J T OWENS
Debugging in a time-sharing environment
AFIPS Proc FJCC Vol 33 1968 7-14
20 A D KARUSH
The computer system recording utility: application and theory
SDC Doc SP-3303 Feb 1969
21 A D KARUSH
Benchmark analysis of time-sharing system
SDC Doc SP-3343 April 1969

APPENDIX A: Advanced development prototype system block diagram.